User Manual for DatatoolTk

Nicola L.C. Talbot
dickimaw-books.com

Version 1.9.20240825 2024-08-25

Contents

List of Figures

1 Introduction

The LaTeX datatool package is able to save databases in its own internal format to allow for rapid loading (using `\DTLwrite[format=dbtex]{⟨file⟩}` with datatool v3.0 or, for earlier versions, `\DTLsaverawdb{⟨file⟩}` or `\DTLprotectedsaverawdb{⟨file⟩}`). Files in this dbtex format are difficult to edit, but they are by far the fastest way of loading a datatool database in LaTeX. The datatooltk application provides a graphical user interface (GUI) to make it easier to edit these files.

DatatoolTk can also import data from csv files, from Excel xls (but not xlsx) or Open Document ods spreadsheets, from structured query language (SQL) databases (only MySQL is currently supported) and from probsoln databases. This manual assumes the user has some knowledge of the datatool package. Please ensure you have at least version 2.15 of datatool installed in your TeX distribution. (Although the latest version is recommended.) Note that datatool v3.0+ is required to support DBTEX 3.0 or DTLTEX 3.0 file formats.

The datatooltk application can be run in either batch mode (default, see §2) or GUI (see §3).

The LaTeX document compilation tool arara (available on CTAN[a]) comes with a datatooltk directive if DatatoolTk is required as part of the document build.

[a] ctan.org/pkg/arara

1.1 What it isn't

The datatooltk application isn't intended to have the full functionality of a spreadsheet or the flexibility of SQL. If you have a large source of data and only require a subset of that data in your LaTeX document, then the most efficient method is to store the data in a SQL database and import it with a SELECT statement that performs the filtering. This may be done in batch mode as part of the document build. It's also possible to filter data imported from a spreadsheet or comma-separated values (CSV) file, but the entire file still needs to be read.

1.2 DBTEX and DTLTEX Files

Supported file formats for both input and output files are: DBTEX 2.0, DBTEX 3.0, DTLTEX 2.0 and DTLTEX 3.0. These formats can be both input and saved in a LaTeX document using datatool's `\DTLread` and `\DTLwrite` commands. (The formats are documented in the user manual for datatool v3.0+ but they are essentially LaTeX files designed for use with datatool.) Note the datatool v3.0+ is required with DBTEX 3.0 and DTLTEX 3.0. Older versions of datatool only support DBTEX 2.0 and DTLTEX 2.0.

> As from datatool v3.0, `\DTLsaverawdb{⟨file⟩}` has been deprecated in favor of `\DTLwrite[format={dbtex-2}]{⟨file⟩}`.

Although the DBTEX and DTLTEX files are all LaTeX files, the file extensions are assumed to be `dbtex` and `dtltex` to reduce the chances of accidentally overwriting the main document file.

In batch mode, an input file can be specified with the `--in` switch, or by simply providing the file name without a preceding switch. In GUI mode, an input file can be opened with the **File ➜ Open...** menu item. Only `dbtex` and `dtltex` files can be input. For other file types, you will need to use the relevant import option.

In batch mode, an output file must be specified with the `--output` (or `-o`) switch. In GUI mode, a file can be saved with the **File ➜ Save** or **File ➜ Save As...** menu items. The output format can be specified with the `--output-format` switch or by selecting the appropriate file filter in the **File ➜ Save As...** file selector. If you use **File ➜ Save**, the format will match the input file.

The output file encoding will default to Java's default encoding, which usually matches the operating system's default, but can be changed with the Java `file.encoding` property (see the Java documentation for further details). Alternatively, you can use the `--tex-encoding` switch or the selector in the **TeX I/O** tab of the **Preferences** dialog box (see §7.4).

DatatoolTk will parse the first line of the input file to see if it matches the format:

```
% ⟨format⟩ ⟨encoding⟩
```

where ⟨*format*⟩ is either `DBTEX` or `DTLTEX` and ⟨*encoding*⟩ is either the canonical encoding name (such as UTF-8) or the `inputenc` label (such as `utf8`). If the first line doesn't match this format, DBTEX 2.0 is assumed with `datatooltk`'s default encoding (see `--tex-encoding` and §7.4).

The `datatooltk` input function will first try parsing the file according to the detected format. If this fails, `datatooltk` will retry using the TeX Parser Library, which has some limited understanding of TeX syntax and `datatool` commands. This is more sophisticated than a simple pattern match, but it's slower. For example, suppose the file contains:

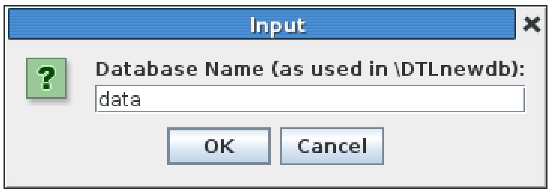
```
\DTLnewdb{testdata}
\DTLnewrow{testdata}
\DTLnewdbentry{testdata}{Info}{sample}
\DTLnewdbentry{testdata}{Value}{1.23}
\DTLnewrow{testdata}
\DTLnewdbentry{testdata}{Info}{another sample}
\DTLnewdbentry{testdata}{Value}{3.75}
```

This doesn't match the dbtex format, so datatooltk will switch to using the TeX Parser Library, which can interpret these commands, but note that if the file is subsequently saved **it will be saved in the dbtex format**. Since this is potentially dangerous (as there's a possibility that you might accidentally load your document tex file) if the TeX Parser Library is used to load the file then, in GUI mode, the filename for that database will be unset, which will trigger the **Save As...** dialog if you try to save the database.

The datatool database files loaded and saved by datatooltk are just LaTeX files, so they could simply have the standard tex extension, but to help differentiate the database files from other files containing TeX/LaTeX code (such as picture-drawing code), datatooltk assumes a default extension of dbtex. If you use this extension, remember to include it in the argument of `\input` or `\DTLloaddbtex`.

To guard against accidentally overwriting a document file, datatooltk now forbids the tex extension for output files.

Note that the database label (as used in commands like `\DTLnewdb`) is independent of the file name (although when importing data, it defaults to the file base name). The database label can be changed using **Edit ➜ Edit Database Name...** in GUI mode or via the command line option `--name` ⟨*label*⟩.


Input

Database Name (as used in \DTLnewdb):

data

Figure 1.1: Setting the Database Name

Example 1: Loading and Displaying Data in the Document

Suppose you have a database file called `my-data.dbtex` and you have set the database label to just "`data`" (as shown in Figure 1.1). Then you can load and display the data using:

```
\documentclass{article}
\usepackage{datatool}% remember to load the datatool package

\input{my-data.dbtex}% load the database from file `my-data.dbtex'

\begin{document}
\DTLdisplaydb{data}
```

```
% Display the database identified by the name `data'
\end{document}
```

If you can't remember the name you assigned to the database, you can access it using \dtl-lastloadeddb.

```
\documentclass{article}
\usepackage{datatool}% remember to load the datatool package

\input{my-data.dbtex}% load the database from file `my-data.dbtex'

\begin{document}
\DTLdisplaydb{\dtllastloadeddb}% Display the last loaded database
\end{document}
```

Alternatively, as from datatool version 2.20, use \DTLloaddbtex instead of \input:

```
\documentclass{article}
\usepackage{datatool}% remember to load the datatool package

\DTLloaddbtex{\mydata}{my-data.dbtex}
% load the database from file `my-data.dbtex'

\begin{document}
\DTLdisplaydb{\mydata}% Display the database
\end{document}
```

If you have at least version 3.0 of datatool, you can use \DTLread instead of \input or \DTLloaddbtex but remember to set the format option to dbtex:

```
\DTLread[format=dbtex]{my-data}
% load the database from file `my-data.dbtex'
```

With DBTEX v2.0, you can't change the database name in the document, but you can with DBTEX v3.0:

```
\documentclass{article}
\usepackage{datatool}% v3.0+
\DTLsetup{default-name=mydata}
\DTLread[name=mydata,format=dbtex]{my-data}

\begin{document}
\DTLaction{display}% Display the database
\end{document}
```

1.3 Verbatim

Since the contents of the database are stored in a TEX token register, and assigned to control sequences via commands like \DTLforeach, verbatim text is not permitted. This is a common problem when attempting to use verbatim text within a command and is covered in the UK List of TEX Frequently Asked Questions (Why doesn't verbatim work within...?). The datatooltk application checks for verbatim text when you load a database or import data (unless the "map TEX special characters" property is set for CSV or SQL imports). Also, datatooltk checks for verbatim text when you edit the contents of a cell. If it detects any, it will give a warning. If you ignore the warning, TEX will give an error if you then attempt to load the database into a document.

ℹ

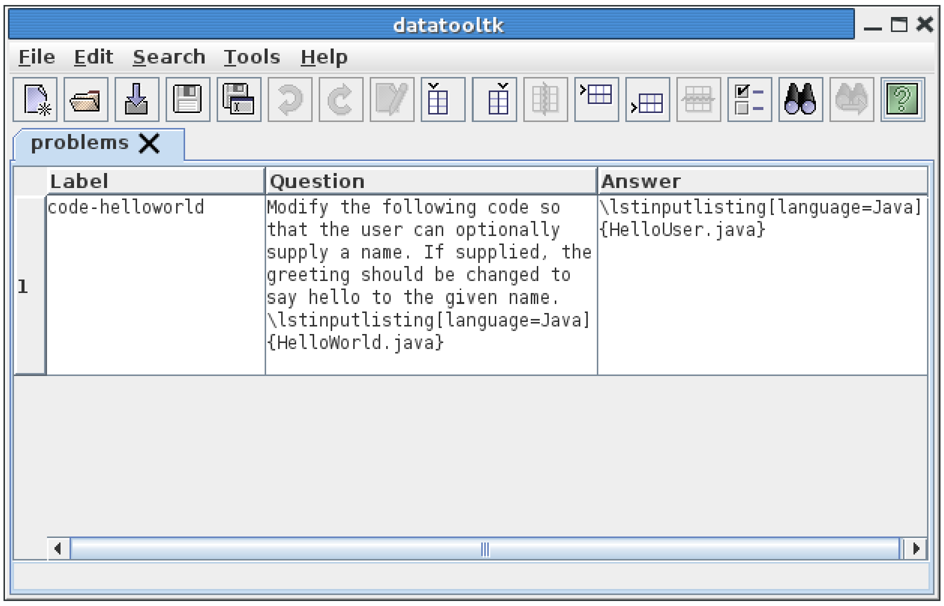
The verbatim check consists of checking for any occurrences of \verb, \lstinline or the beginning of the verbatim, lstlisting or alltt environments. Other verbatim-like commands or environments are not recognised.

If you just have a short fragment of inline verbatim text, consider one of the alternatives listed in the FAQ. If on the other hand you have a block of verbatim text you'll have to put the verbatim text in a separate file and load it using \verbatiminput (from the verbatim package) or \lstinputlisting (from the listings package). For example, in Figure 1.2 I have used \lstinputlisting.

That database requires two files: HelloWorld.java

```
public class HelloWorld
{
   public static void main(String[] args)
   {
      System.out.println("Hello World!");
   }
}
```

and HelloUser.java:



	Label	Question	Answer
1	code-helloworld	Modify the following code so that the user can optionally supply a name. If supplied, the greeting should be changed to say hello to the given name. \lstinputlisting[language=Java]{HelloWorld.java}	\lstinputlisting[language=Java]{HelloUser.java}

Figure 1.2: Verbatim Blocks Need to be in Separate Files

```
public class HelloUser
{
   public static void main(String[] args)
   {
      System.out.println("Hello "
         + (args.length==0 ? "anon" : args[0])+"!");
   }
}
```

Assuming that I've saved my database in a file called `prob-verb.dbtex` with database label "`problems`", here's a sample document:

```
\documentclass{article}

\usepackage{etoolbox}
\usepackage{datatool}
\usepackage{listings}

\newtoggle{showanswers}
\toggletrue{showanswers}

\input{prob-verb.dbtex}

\begin{document}
```

```
\begin{enumerate}
  \DTLforeach*{problems}{\Question=Question,\Answer=Answer}%
  {%
    \item \Question

    \iftoggle{showanswers}{Answer: \Answer}{}
  }
\end{enumerate}

\end{document}
```

see also:

- 4.2 Shuffling the Data
- 4.3 Sorting and Shuffling
- 5.4 Import probsoln Data

1.4 Leading/Trailing Spaces

Spaces characters at the start and end of cell contents and column titles are removed when writing the dbtex file (but they will still show in the GUI until the file is saved and reloaded).

Leading space characters are naturally ignored in datatool's internal format. For consistency, datatooltk now trims leading and trailing spaces from each cell and column title when writing the dbtex file as they are usually unwanted and easy to miss. If you explicitly want a space you need to use LaTeX markup, such as \space. A trailing space can also be hidden by a final comment. For example, with the cell contents set to:

```
\space text % comment
```

then there will be a leading and trailing space if the value is used in the document. Any space characters at the end of the comment line will be stripped, but those would naturally be ignored by LaTeX anyway. Remember that blank lines are converted to \DTLpar, so paragraph breaks won't be recognised as spaces by the trimming code. Column and database ⟨*labels*⟩ will only have leading and trailing spaces stripped if the auto trim labels setting is on. Column titles are always trimmed when writing the dbtex file.

1.5 Null Values

Empty entries aren't the same as null entries. If you want a null entry, set the entry to \@dtl-novalue. A convenient way to do this is to select the cell and use **Edit ➜ Set Cell to Null**. Alternatively, you can set all entries in a selected column to null with **Edit ➜ Column ➜ Nullify Column** and similarly for a selected row with **Edit ➜ Row ➜ Nullify Row**.

In your LaTeX document, you can check for null values using `datatool`'s `\DTLifnull` command. To check for empty values you can use one of `etoolbox`'s commands, such as `\ifdefempty`. As from `datatool` version 2.20, you can also use `\DTLifnullorempty`.

2 Command Line Options

The DatatoolTk application may be run from the command line in batch mode:

```
datatooltk [⟨options⟩]
```

In this case, a load or import option must be used (such as --in or --csv), as well as an output file specified with --output. The application may also be run with a graphical interface:

```
datatooltk-gui [⟨options⟩]
```

This is essentially equivalent to

```
datatooltk --gui [⟨options⟩]
```

but additionally has a splash screen. In GUI mode, the output file should not be specified, and the load/import switches may be omitted. The **File** menu may be used to load, import and save files.

Available options for the command line invocation are listed below. Note that some of the default values may be changed through the GUI, which saves your preferences. These will be picked up by the next batch invocation.

You can't combine any of the load/import options: --in, --csv, --xls, --ods, --sql, --probsoln. You also can't combine any of the merge options: --merge, --merge-csv, --merge-xls, --merge-ods, --merge-sql, --merge-probsoln. The merge import options use the same settings as the import options. If you want to merge, for example, sheet 1 and sheet 2 from the same spreadsheet, you will have to first import one or both of them to a dbtex file and then perform the merge. With the exception of --merge-sql, all the merge options will ignore a missing file and just print a warning to STDERR.

The import functions are one-way. You can't export back to any of those formats.

--gui (or -g)

Invoke datatooltk in GUI mode.

--batch (or -b)

Invoke `datatooltk` in batch mode (default).

```
--output ⟨filename⟩ (or -o)
```

Save the database to ⟨*filename*⟩ (batch mode only). To guard against accidentally overwriting a document file, `datatooltk` now forbids the `tex` extension for output files. See §1.2.

```
--output-format ⟨format⟩
```

Sets the default file format for output files. The value may be one of (case-insensitive): "`DBTEX 2.0`", "`DBTEX 3.0`", "`DTLTEX 2.0`", "`DTLTEX 3.0`". The value may also be an empty string or "`default`" to indicate the default format should be used. The default will either match the format of the input file or, if data was imported, "`DBTEX 3.0`". Alternatively, this setting can be saved in the **TeX I/O** tab of the **Preferences** dialog box (see §7.4).

If the DBTEX 3.0 format is used, there are further settings to determine whether or not to store the entry value in `datatool`'s datum format. See §7.4 for these settings.

```
--in ⟨filename⟩ (or -i)
```

Load ⟨*datatool file*⟩. The switch `--in` (or `-i`) is optional, so:

```
datatooltk ⟨file⟩
```

is equivalent to

```
datatooltk --in ⟨file⟩
```

See §1.2 for permitted file types.

```
--tex-encoding ⟨encoding⟩
```

Set the encoding for the TeX (`tex`, `dtltex` and `dbtex`) files to ⟨*encoding*⟩. The ⟨*encoding*⟩ may be the keyword "`default`" or an empty string to indicate Java's default file encoding (see §1.2). Make sure that your LaTeX document matches the given ⟨*encoding*⟩. The encoding of CSV files is independent of the TeX (`dbtex`) encoding and is set through `--csv-encoding`.

`--name` ⟨*label*⟩

If used with `--in`, `--csv`, `--sql`, `--xls`, `--ods` or `--probsoln`, the `--name` switch sets the database label to ⟨*name*⟩. (See §1.2.)

`--version` (or `-v`)

Print the version details to STDOUT and exit.

`--help` (or `-h`)

Print a brief summary of available options to STDOUT and exit.

`--debug`

Enable debug mode.

`--nodebug`

Disable debug mode. (Default.)

`--compat` ⟨*level*⟩

⟨*level*⟩ Set the compatibility level. The argument may be the keyword " `latest` " (no backward compatibility required) or " `1.6` " (compatible with version 1.6 and below). The `--compat 1.6` setting only affects `--shuffle` and is provided for old documents.

`--owner-only`

Set read/write permissions when saving `dbtex` files to owner only. (Has no effect on some operating systems.)

`--noowner-only`

Don't change read/write permissions when saving `dbtex` files.

`--map-tex-specials`

Map TeX special characters when importing data from CSV or SQL.

```
--nomap-tex-specials
```

Don't map TeX special characters when importing data from CSV or SQL. (Default.)

```
--auto-trim-labels
```

Automatically strip leading and trailing spaces from database and column identifiers. (See also §1.4.)

```
--noauto-trim-labels
```

Don't automatically strip leading and trailing spaces from database and column identifiers.

`--seed` ⟨*number*⟩

Set the random generator seed to ⟨*number*⟩ or clear it if ⟨*number*⟩ is the empty string `""`. (See §4.2.)

```
--shuffle
```

Shuffle the database. (Shuffle is always performed after sort, regardless of the option order.)

```
--noshuffle
```

Don't shuffle the database. (Default.)

`--sort` [⟨*prefix*⟩]⟨*field*⟩

Sort the database according to the column whose label is ⟨*field*⟩. Optionally, ⟨*prefix*⟩ may be the plus character "`+`" (ascending order) or the hyphen/minus character "`-`" (descending order). If ⟨*prefix*⟩ is omitted, ascending is assumed. (See §4.1.)

`--sort-locale` ⟨*value*⟩

If the ⟨*value*⟩ is the keyword "`none`" use letter-sorting for strings. That is, compare the Unicode values of each character. Otherwise ⟨*value*⟩ should be a valid internet engineering task force (IETF) language tag that identifies a locale. Strings will then be sorted according to that locale's alphabetical order. Note that `datatooltk` can't interpret LaTeX commands. (If you need that ability, you might want to consider using `bib2gls` with `glossaries-extra` instead.) The default setting is to use a letter-sort for strings.

This setting also governs the string comparison functions used by the filter option `--filter`.

```
--sort-case-sensitive
```

(Ignored with locale-sensitive comparisons.) Use case-sensitive comparison when letter-sorting strings. This setting also governs the string comparison functions used by the filter option `--filter`.

```
--sort-case-insensitive
```

(Default.) Use case-insensitive comparison when letter-sorting strings. This compares the lower case versions of the strings using a character code comparison. This setting also governs the string comparison functions used by the filter option `--filter`.

`--truncate` ⟨*n*⟩

Truncate the database to the first ⟨*n*⟩ rows. (Has no effect if ⟨*n*⟩ is greater than or equal to the total number of rows.) Truncation is always performed after any sorting, shuffling and filtering, but before column removal.

`--remove-columns` ⟨*column list*⟩

Remove the columns identified by ⟨*column list*⟩, which may be a comma-separated list of column labels (for example, "`Details,Comments`") or a comma-separated list of column indexes or ranges (where the first column has the index 1). For example, "`3,5-7,9`" indicates columns 3, 5, 6, 7 and 9. You can't mix labels and indexes, so "`Details,5,Comments`" would mean the columns identified by the labels "`Details`" "`5`" (which might not correspond to the fifth column) and "`Comments`". Ranges are only permitted with indexes and may be open ended. For example, `-4` is equivalent to `1-4` and `3-` means the third column onwards. This option is not cumulative and may not be used with `--remove-except-columns`. Column removal is always performed last, so you can still use `datatooltk` to sort or merge by a column that isn't required in the document.

`--remove-except-columns` ⟨*column list*⟩

This option is similar to `--remove-columns` but the ⟨*column list*⟩ indicates which columns to keep. All other columns are removed. The argument has the same syntax as for `--remove-columns`. This option is not cumulative and may not be used with `--remove-columns`. Column removal is always performed last, so you can still use `datatooltk` to sort or merge by a column that isn't required in the document.

`--filter` ⟨*key*⟩ ⟨*operator*⟩ ⟨*value*⟩

Adds the given filter. This filter returns true if the value in the column whose label is given by ⟨*key*⟩ matches the operation ⟨*operator*⟩ ⟨*value*⟩ where ⟨*operator*⟩ may be one of:

- `eq` (equals ⟨*value*⟩),
- `ne` (does not equal ⟨*value*⟩),
- `le` (less than or equal to ⟨*value*⟩),
- `lt` (less than ⟨*value*⟩),
- `ge` (greater than or equal to ⟨*value*⟩),
- `gt` (greater than ⟨*value*⟩),
- `regex` (matches the regular expression ⟨*value*⟩).

Multiple filters may be used. The regular expression should be in the format used by java.util.regex.Pattern. Filtering is always performed after sorting and shuffling. Numerical comparisons are used for columns that are identified as either integer or real data types otherwise string comparisons are used, except in the case of `regex` where the data type is disregarded and all values are assumed to be strings. (If the column type is identified as having an integer or real data type but ⟨*value*⟩ is not a number, a string comparison will be used.) For example: `--filter Level eq 3` means that the filter should return true if the value in the column whose label is "`Level`" is equal to 3. If there isn't a column with the label ⟨*key*⟩, a warning is printed on the standard error stream and the filter is ignored.

`--filter-or`

Use OR operator when filtering. (Default.) This has no effect if you only supply one filter.

`--filter-and`

Use AND operator when filtering. This has no effect if you only supply one filter.

`--filter-include`

When filtering, discard rows that don't match the filter (and keep those that do match). This is the default action.

`--filter-exclude`

When filtering, discard rows that match the filter (and keep those that don't match).

`--merge` ⟨*key*⟩ ⟨*db file*⟩

Merges the input or imported database with the database stored in the given ⟨*db file*⟩. Each row in ⟨*db file*⟩ is merged with the row that has a matching value in the column whose label is given by ⟨*key*⟩. Both databases must contain a column with that label. (Each entry in that column should ideally be unique.) If no matching row is found, a new row is added. If both databases share additional columns, the values in ⟨*db file*⟩ override those in the original database. If ⟨*db file*⟩ doesn't exist, a warning is issued and the option is ignored. This option is always implemented before any sorting, shuffling, filtering or truncating. Only one of the merge options is permitted.

`--csv` ⟨*csv file*⟩

Import data from the given CSV file. (See §5.1.)

`--merge-csv` ⟨*key*⟩ ⟨*csv file*⟩

As `--merge` but the data to be merged is imported from the given CSV file.

`--csv-sep` ⟨*character*⟩

Specify the character used to separate values in the CSV file. (Defaults to a comma (`,`) character.)

`--csv-delim` ⟨*character*⟩

Specify the character used to delimit values in the CSV file. (Defaults to a double-quote (`"`) character.)

`--csv-skiplines` ⟨*n*⟩

Skip the first ⟨*n*⟩ rows of the `--csv` file. (Useful if you have a comment block at the start of the file that needs to be skipped.) The value ⟨*n*⟩ may be 0 (don't skip) or a positive integer indicating the number of rows to skip. Blank rows are always included in this count, even if `--csv-skip-empty-rows` is set. The spreadsheet import functions also use this setting.

`--csv-strictquotes`

Ignore any undelimited information (where the delimiter is given by `--csv-delim`).

`--nocsv-strictquotes`

Allow undelimited data.

`--csv-encoding` ⟨*encoding*⟩

Set the encoding for the CSV files to ⟨*encoding*⟩. The ⟨*encoding*⟩ may be the keyword “`default`” or an empty string to indicate the default. The encoding of the TEX (`dbtex`) file is independent of the CSV encoding and is set through `--tex-encoding`.

`--csv-header`

The CSV file has a header row. (Default.) The spreadsheet import functions also use this setting.

`--nocsv-header`

The CSV file doesn't have a header row. The spreadsheet import functions also use this setting.

`--csv-skip-empty-rows`

Skip empty lines found in the CSV file. (Spreadsheet import also uses this setting.)

`--nocsv-skip-empty-rows`

Don't skip empty lines found in the CSV file. (Spreadsheet import also uses this setting.)

`--csv-escape` ⟨*character*⟩

Set the CSV file escape character to ⟨*character*⟩. If your data includes the delimiter character, you need to escape that character to prevent it from being mistaken for the delimiter. For example, if the delimiter is the double-quote (") character and the escape character is the backslash (\ ) character, then a row of data may appear as:

```
12345,"A \"sample\" entry."
```

(This won't actually render properly in LATEX as won't produce the typographically correct double quotes.) By default the CSV escape character is the backslash (\ ) character (as in the above example) which means that you must double the backslash if you have any (La)TeX commands within the file. To avoid this, you can set the escape character to something else that doesn't occur in your data.

Note that delimiter character is often also the escape character, so the row of data would be:

```
12345,"A ""sample"" entry."
```

`--nocsv-escape` ⟨*character*⟩

Don’t have an escape character for your CSV file.

`--xls` ⟨*xls file*⟩

Import data from the given Excel `xls` file. (See §5.2.)

`--merge-xls` ⟨*key*⟩ ⟨*xls file*⟩

As `--merge` but the data to be merged is imported from the given Excel `xls` file.

`--ods` ⟨*ods file*⟩

Import data from the given Open Document Spreadsheet `ods` file. (See §5.2.)

`--merge-ods` ⟨*key*⟩ ⟨*ods file*⟩

As `--merge` but the data to be merged is imported from the given Open Document Spreadsheet `ods` file.

`--sheet` ⟨*sheet id*⟩

The sheet to select from the Excel workbook or Open Document Spreadsheet. This may either be an index (starting from 0) or the name of the sheet. If this option is omitted, the first sheet is assumed.

`--sql` ⟨*query statement*⟩

Import data from an SQL database where ⟨*statement*⟩ is an SQL SELECT statement. (See §5.3)

`--merge-sql` ⟨*key*⟩ ⟨*query statement*⟩

As `--merge` but the data to be merged is imported using the given SQL SELECT statement.

`--sqldb` ⟨*name*⟩

The SQL database name.

`--sqlprefix` ⟨*prefix*⟩

The Java SQL prefix. (Default: "`jdbc:mysql://`".) Currently, only MySQL is supported. Additional libraries will be required for other SQL databases.

`--sqlport` ⟨*port*⟩

The SQL port number. (Default: 3306.)

`--sqlhost` ⟨*port*⟩

The SQL host. (Default: "`localhost`".)

`--sqluser` ⟨*username*⟩

The SQL user name.

`--sqlpassword` ⟨*password*⟩

The SQL password (insecure). If omitted, you will be prompted for a password if you try to import data from an SQL database.

`--wipepassword`

For extra security, wipe the password from memory as soon as it has been used to connect to an SQL database. (Default.)

`--nowipepassword`

Don't wipe the password from memory as soon as it has been used to connect to an SQL database.

`--noconsole-action` ⟨*action*⟩

If in batch mode and a SQL password is required and `--sqlpassword` hasn't been used, the default action is for `datatooltk` to request a password via the console. If there is no console available the action is determined by ⟨*action*⟩ which may be one of:

- "`error`" issue an error;

- “ `stdin` ” request the password via the standard input stream (less secure than using a console, and can produce an annoying flicker);
- “ `gui` ” display a dialog box in which to enter the password (default).

`--probsoln` ⟨*filename*⟩

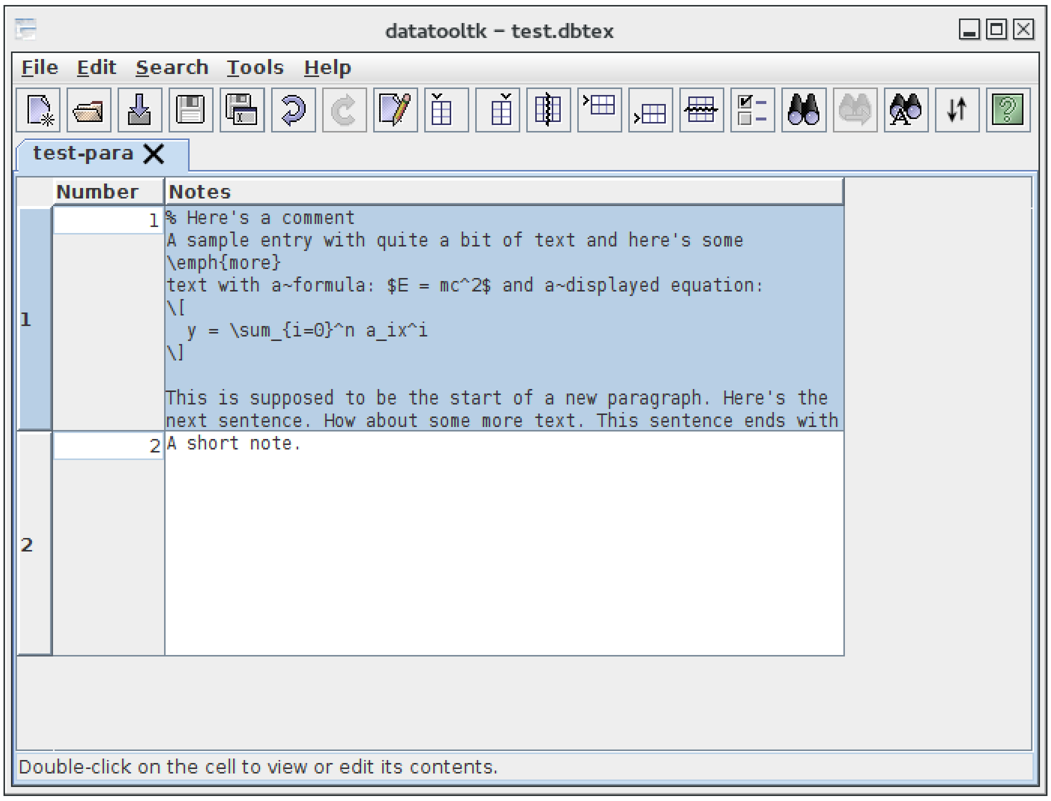
Import `probsoln` data from ⟨*filename*⟩. (See §5.4.)

`--merge-probsoln` ⟨*key*⟩ ⟨*filename*⟩

As `--merge` but the data to be merged is imported from the given `probsoln` data set file.

3 Graphical Mode

To run datatooltk in graphical mode you must invoke it with either datatooltk-gui or datatooltk --gui. The main window is shown in Figure 3.1. Each database is in a tabbed pane, with the name of the database in the tab. Note that the name corresponds to the database's identifying label, as used in commands like \DTLnewdb. This is not necessarily the same as the filename (see §1.2). Since this name is used as a label, it shouldn't contain any of TeX's special characters or any other active characters that could cause problems. An asterisk * following the label in the tab indicates that the database has been modified. If you move the mouse over the tab, you will see the full pathname appear in a tooltip, if the database has been saved to a datatool file, and the filename (without the path) will be shown in the title bar.



```
1 % Here's a comment
  A sample entry with quite a bit of text and here's some
  \emph{more}
  text with a~formula: $E = mc^2$ and a~displayed equation:
  \[
    y = \sum_{i=0}^n a_ix^i
  \]

  This is supposed to be the start of a new paragraph. Here's the
  next sentence. How about some more text. This sentence ends with
2 A short note.
```

Double-click on the cell to view or edit its contents.

Figure 3.1: Main Window

You can use the **File** menu to create a new database, load an existing database or import data (see §5). To load an existing database, use **File ➜ Open...**. These database files contain LaTeX code in a specific format. The datatooltk application assumes a dbtex file extension (see §1.2). You can load these files into a LaTeX document using \input or \DTLloaddbtex, but remember to specify the dbtex extension. (Also remember to load the datatool package.)

Each column has a corresponding data type: string, integer, real or currency. The type is automatically detected from the column data, but can be changed, as described in §3.2.

Non-string entries can be edited by double-clicking on the relevant cell, or you can select the relevant cell and use **Edit ➜ Edit Cell…**. In the first case, a cursor will appear in the cell and you can edit the numerical value and press "Enter" to finish editing. In the second case, the cell editor dialog box will open, see §3.1.

Only the first few lines of a string entry are visible in the main window. If an entry has more than that number of lines, you will need to use the cell editor dialog box to view the entire contents of that cell.

The default row height can be changed in the **Preferences** dialog box (see §7). Columns set to integer or real data types have single-lined cells with no line wrap. Columns set to currency data type may wrap, but using Enter will finish editing the cell (unless you're using the cell editor dialog box). If you insert a newline character in the cell edit dialog box (for any data type), the type for that column will be converted to "string".

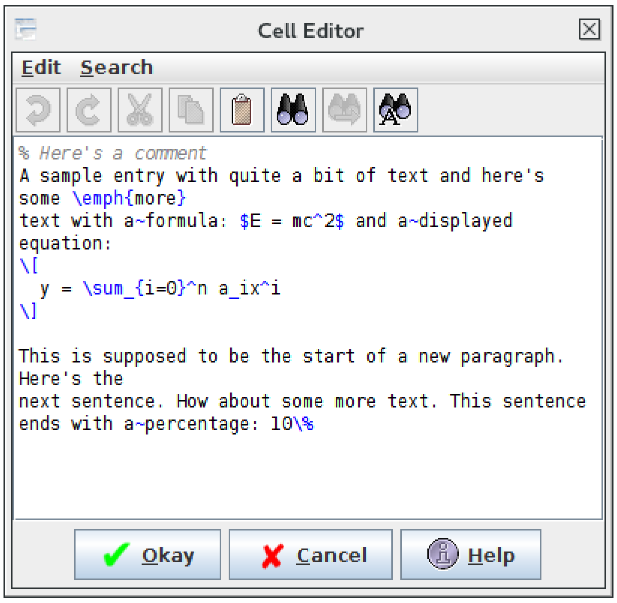
To edit or view an entry in a column with the "string" data type, double-click on the relevant cell or select the cell and use **Edit ➜ Edit Cell…** to open the cell editor dialog box (see §3.1). You can now scroll through the cell contents.

3.1 Cell Editor

To open the cell editor dialog box (see Figure 3.2) double-click on the required cell, which must be in a column with a string data type. Alternatively, select the cell (of any type) and use **Edit ➜ Edit Cell…**.

Remember that the contents of the cell should be LaTeX code, so be careful if you use any of TeX's special characters. Also, see the section on verbatim text (§1.3) if you haven't already read it. Once you have made your edits, click on **Okay** to update the database. To discard the edits, click **Cancel**.

If you've used datatool, you will probably know that if you want a paragraph break in your cell entries you need to use \DTLpar, but with datatooltk you don't need to worry about it as blank lines in an entry will automatically be converted behind the scenes. Note that redundant blank lines will be removed. Leading and trailing spaces are ignored when writing the dbtex file, but they will still be present in the cell editor until the file is saved and reloaded (see §1.4).



```
% Here's a comment
A sample entry with quite a bit of text and here's
some \emph{more}
text with a~formula: $E = mc^2$ and a~displayed
equation:
\[
  y = \sum_{i=0}^n a_ix^i
\]

This is supposed to be the start of a new paragraph.
Here's the
next sentence. How about some more text. This sentence
ends with a~percentage: 10\%
```

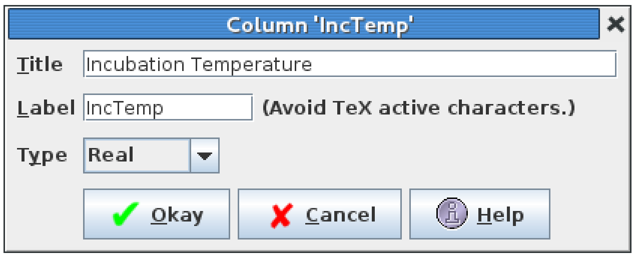
Figure 3.2: Cell Editor Dialog

If you use `datatool`'s `\DTLwrite`, `\DTLsaverawdb` or `\DTLprotectedsaverawdb` commands to overwrite your file, you will lose any pretty-printing spaces or comments in your code.

3.2 Header Dialog

Each column has a title, a uniquely identifying label and an associated type. The type can be one of: **String**, **Integer**, **Real** or **Currency**. The type is automatically detected from the column data, but can be changed using the **Edit** ➜ **Column** ➜ **Edit Header** menu item or by double-clicking on the column header which opens the header dialog box (see Figure 3.3). The label corresponds to the label used to identify the column in commands such as `\DTLforeach` and will be trimmed if the `--auto-trim-labels` setting is on. The title is used in commands like `\DTLdisplaydb`. See §7 for currency mappings. If the title field is left blank, it will be assigned the same value as the label.

In GUI mode, column headers show the title. If you move the mouse over the column header, you will see the label and type displayed in a tooltip (see Figure 3.4).

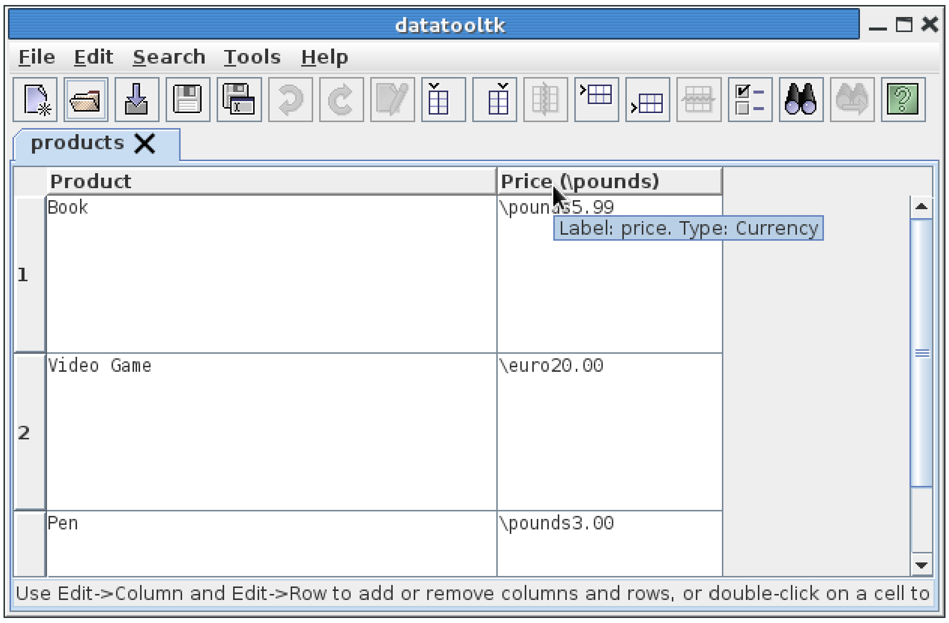


Title: Incubation Temperature

Label: IncTemp (Avoid TeX active characters.)

Type: Real

Figure 3.3: Header Dialog



	Product	Price (\pounds)
1	Book	\pounds5.99
2	Video Game	\euro20.00
	Pen	\pounds3.00

Label: price. Type: Currency

Use Edit->Column and Edit->Row to add or remove columns and rows, or double-click on a cell to

Figure 3.4: Header Details Shown in Tooltip

4 Tools

There are currently two tools available: sort (see §4.1) and shuffle (see §4.2). These both reorder the rows of the database and can be invoked either from the **Tools** menu or from the command line (as long as you have also loaded a database using `--in` or one of the import options). If you use both `--sort` and `--shuffle` in the command line invocation, sort will always be performed first, regardless of the option order.

4.1 Sorting the Data

Although you can sort data in `datatool` using `\DTLsort`, it's far more efficient to sort it in `datatooltk`. So instead of doing, say,

```
\input{mydata.dbtex}
% loads database labeled `data' from file `mydata.dbtex'
\DTLsort{Surname}{data}% sort data on `Title' field
% Later in the document:
\DTLdisplaydb{data}% display data in tabular environment
```

It's better to run, say,

```
datatooltk --in mydata.dbtex --sort Surname --output
mydata-sorted.dbtex
```

(Remember that this defaults to letter sorting for strings. Use `--sort-locale` to sort according to a locale.) Then in the document, just load `mydata-sorted.dbtex`:

```
\input{mydata-sorted.dbtex}
% Later in the document:
\DTLdisplaydb{data}% display data in tabular environment
```

or, if you have the shell escape enabled you can used TEX's `\write18` mechanism:

```
\immediate\write18{datatooltk --in
mydata.dbtex --sort Surname
--output mydata-sorted.dbtex}

\input{mydata-sorted.dbtex}
% Later in the document:
\DTLdisplaydb{data}% display data in tabular environment
```

Beware of the security implications of enabling the shell escape.

If you have arara version 4.0+, there's a rule for datatooltk:

```
% arara: datatooltk: {input: mydata.dbtex, sort: Surname,
% arara: --> output: mydata-sorted.dbtex}
% arara: pdflatex
```

Incorporating datatooltk into the document build in this way is safer than enabling the unrestricted shell.

If the original data is in an SQL database, it's even more efficient to do the sorting in the SELECT statement when you import the data (see §5.3).

A database can be sorted according to a particular column in either ascending or descending order. In batch mode, this is done with the `--sort` option, as shown above, where the sort column is identified by the column's unique label. If the label is preceded by `-` then descending order is used (for example, `--sort -Surname`). If the label is preceded by `+` (or has no prefix) then ascending order is used. When comparing strings there are two modes: letter (compare character codes) or locale-sensitive (use the alphabet for the given locale). For letter comparisons you can also use `--sort-case-sensitive` for case-sensitive comparisons and `--sort-case-insensitive` for case-insensitive comparisons. The default is case-insensitive. The locale comparisons are typically case-insensitive. The treatment of accented characters depends on the locale's rule.

In GUI mode, sorting is done using the **Tools ➜ Sort...** menu item which opens the **Sort Database** dialog box (see Figure 4.1).

Select the column you wish to sort by from the drop-down list of column titles, and check the appropriate radio button for ascending or descending sort. If the column has the string data type, you also need to specify what type of comparison you want to use. For a letter (character code) comparison, select the **Character Code** box, which will enable the **Case sensitive** box.

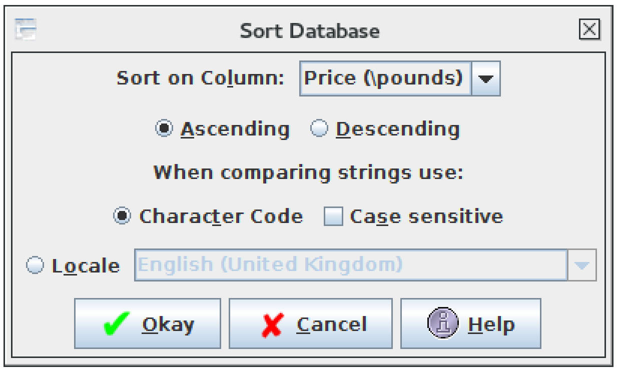

Sort Database
Sort on Column: Price (\pounds)
[✓ Ascending] Descending
When comparing strings use:
[✓ Character Code] Case sensitive
Locale English (United Kingdom)
Okay
Cancel
Help

Figure 4.1: Sort Dialog

For a locale comparison, select the **Locale** box, which will enable the locale selector.

If the column type has a numerical type, the entries will be sorted via a numerical comparison (10 is greater than 2) and the string options are ignored. If the column type is a string type, any numerical entries will be sorted via a character comparison ("10" comes before "2").

Example 2: Sorting Data

Consider the data shown in Figure 4.2 and reproduced in Table 4.1.

Table 4.1: Original Data

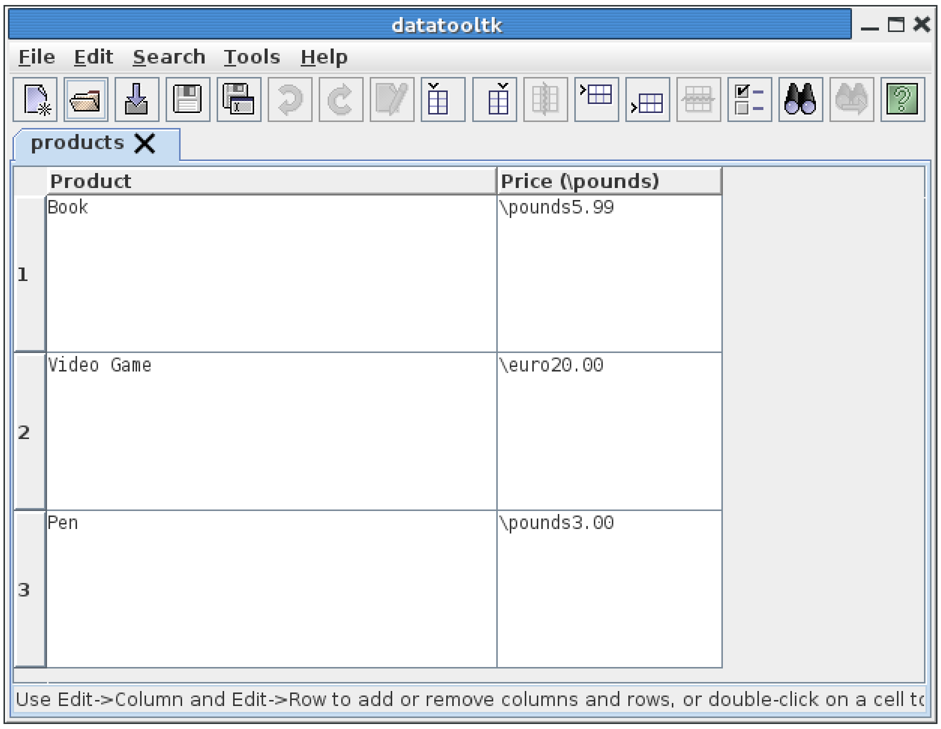
Book	`\pounds5.99`
Video Game	`\euro20.00`
Pen	`\pounds3.00`

The first column has a string data type and the second has a currency data type. Sorting in ascending order on the second column, will sort numerically on just the number. The currency symbol is ignored (see Table 4.2). If the type of the second column is changed from currency to string, and the sort is redone, the order is now based on a string comparison that includes the currency symbol (see Table 4.3).

Table 4.2: Data Sorted on Second Column (Currency Comparison)

Pen	`\pounds3.00`
Book	`\pounds5.99`
Video Game	`\euro20.00`

Remember that `datatooltk` doesn't have any knowledge of currency conversions. In this example it would be better to have a column of real numbers containing the price in a single base currency. (In fact, it would be better to store the original data in a spreadsheet or database and just use `datatooltk` in batch mode.)



	Product	Price (\pounds)
1	Book	\pounds5.99
2	Video Game	\euro20.00
3	Pen	\pounds3.00

Use Edit->Column and Edit->Row to add or remove columns and rows, or double-click on a cell tc

Figure 4.2: Original Data

Table 4.3: Data Sorted on Second Column (String Comparison)

Video Game	`\euro20.00`
Pen	`\pounds3.00`
Book	`\pounds5.99`

4.2 Shuffling the Data

Shuffling involves randomly changing the order of the rows. This can be performed either by the `--shuffle` command line option or the **Tools ➜ Shuffle** menu item. You can change the seed used by the random number generator with `--seed` or through the **Preferences** dialog box (see §7). The method used to shuffle data has changed since version 1.6. If you need to use the old method (for example, if you have set a seed with an existing document), then use `--compat 1.6` when invoking `datatooltk`. For example

```
datatooltk --compat 1.6 --seed 2000 --shuffle infile.dbtex -o
outfile.dbtex
```

The newer version is more efficient.

Example 3: Shuffling Rows

Consider the database shown in Figure 4.3. This database has three columns. The first is a question, the second is the corresponding answer (optional) and the third is a number indicating the question level. For example, 1 could correspond to easy and 2 could correspond to medium difficulty.

Now suppose I want to write an assignment sheet that has one randomly selected question of level 1 and two randomly selected questions of level 2. Let's suppose the file name is `data.dbtex` and the database label is "`problems`". Then I can run `datatooltk` in batch mode using:

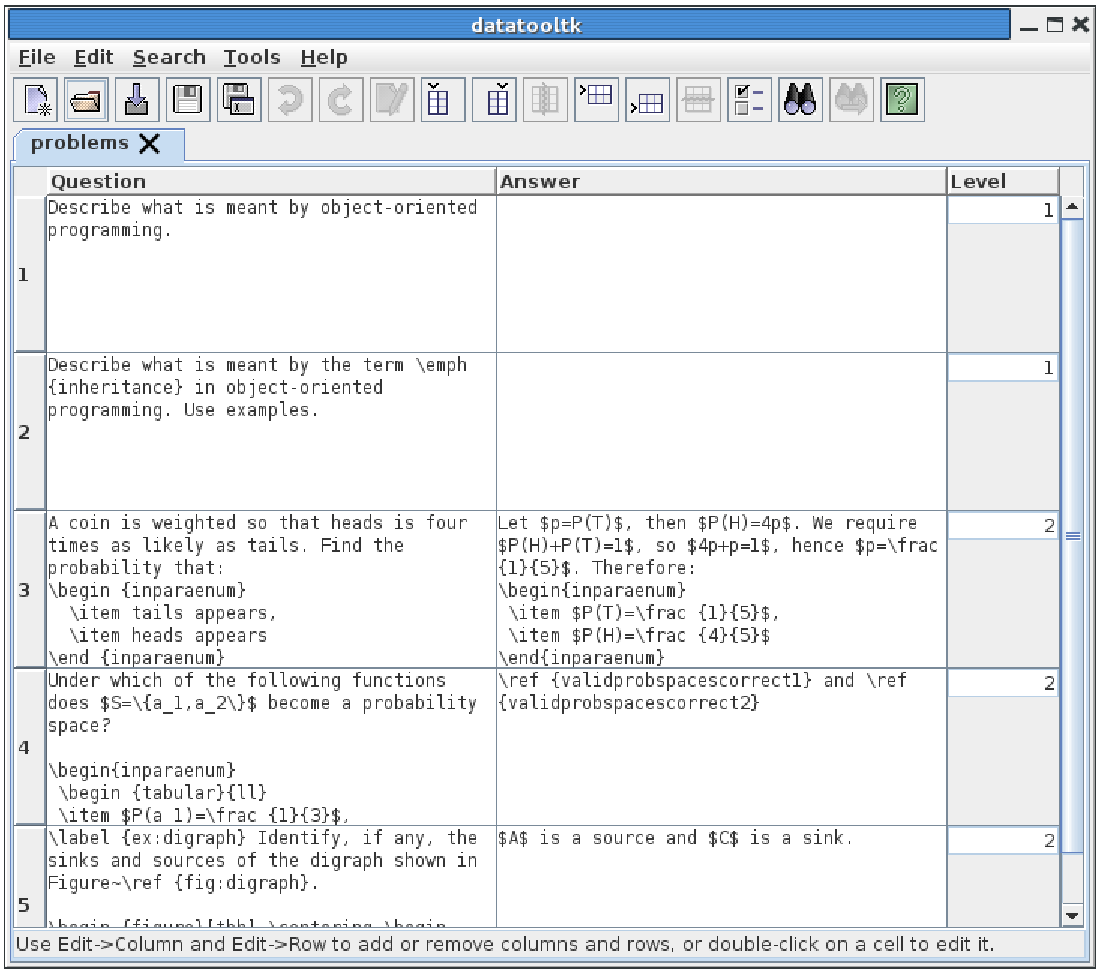
```
datatooltk --shuffle --in data.dbtex --output data-shuffled.dbtex
```

Remember to use `--seed` if you don't want a different ordering every time you run that command. For example:

```
datatooltk --seed 2013 --shuffle --in data.dbtex --output
data-shuffled.dbtex
```

This shuffled database can now be loaded in my document:

```
\documentclass{article}

\usepackage{etoolbox}
\usepackage{datatool}

% Used by some of the questions:
\usepackage{paralist}
\usepackage{tikz}
```



	Question	Answer	Level
1	Describe what is meant by object-oriented programming.		1
2	Describe what is meant by the term \emph {inheritance} in object-oriented programming. Use examples.		1
3	A coin is weighted so that heads is four times as likely as tails. Find the probability that: \begin {inparaenum} \item tails appears, \item heads appears \end {inparaenum}	Let $p=P(T)$, then $P(H)=4p$. We require $P(H)+P(T)=1$, so $4p+p=1$, hence $p=\frac {1}{5}$. Therefore: \begin{inparaenum} \item $P(T)=\frac {1}{5}$, \item $P(H)=\frac {4}{5}$ \end{inparaenum}	2
4	Under which of the following functions does $S=\{a_1,a_2\}$ become a probability space? \begin{inparaenum} \begin {tabular}{ll} \item $P(a_1)=\frac {1}{3}$,	\ref {validprobspacescorrect1} and \ref {validprobspacescorrect2}	2
5	\label {ex:digraph} Identify, if any, the sinks and sources of the digraph shown in Figure~\ref {fig:digraph}. \begin {figure}[tbh] \centering \begin	A is a source and C is a sink.	2

Use Edit->Column and Edit->Row to add or remove columns and rows, or double-click on a cell to edit it.

Figure 4.3: Shuffle Example

```
\newtoggle{showanswers}
\toggletrue{showanswers}

\input{data-shuffled.dbtex}

% Number to select from level 1
\newcounter{maxleveli}
\setcounter{maxleveli}{1}

% Number to select from level 2
\newcounter{maxlevelii}
\setcounter{maxlevelii}{2}

% Counter to keep track of level 1 questions
\newcounter{leveli}

% Counter to keep track of level 2 questions
\newcounter{levelii}

\begin{document}

\begin{enumerate}
 \DTLforeach*{problems}%
  {\Question=Question,\Answer=Answer,\Level=Level}%
  {%
    % Increment counter for this level
    \stepcounter{level\romannumeral\Level}%
    % Have we reached the maximum for this level?
    \ifnumgreater
      {\value{level\romannumeral\Level}}%
      {\value{maxlevel\romannumeral\Level}}%
    {}% reached maximum, do nothing
    {\item \Question

     \ifdefempty\Answer
     {}% no answer
     {% do answer if this is the solution sheet
       \iftoggle{showanswers}{Answer: \Answer}{}%
     }%
    }%
    % do we need to continue or have we got everything?
    \ifboolexpr
    {%
```

```
      test{\ifnumgreater{\value{leveli}}{\value{maxleveli}}}
      and
      test{\ifnumgreater{\value{levelii}}{\value{maxlevelii}}}
    }%
    {\dtlbreak}{}%
  }
\end{enumerate}

\end{document}
```

What if I want all the easy questions listed first? This requires some modifications to the code as shown below:

```
\documentclass{article}

\usepackage{etoolbox}
\usepackage{datatool}

% Used by some of the questions:
\usepackage{paralist}
\usepackage{tikz}

\newtoggle{showanswers}
\toggletrue{showanswers}

\input{data-shuffled.dbtex}

% Number to select from level 1
\newcounter{maxleveli}
\setcounter{maxleveli}{1}

% Number to select from level 2
\newcounter{maxlevelii}
\setcounter{maxlevelii}{2}

% Counter to keep track of level 1 questions
\newcounter{leveli}

% Counter to keep track of level 2 questions
\newcounter{levelii}

% List of level 1 questions
\newcommand*{\listleveli}{}

% List of level 2 questions
```

```
\newcommand*{\listlevelii}{}

\begin{document}

 \DTLforeach*{problems}%
  {\Question=Question,\Answer=Answer,\Level=Level}%
  {%
    % Increment counter for this level
    \stepcounter{level\romannumeral\Level}%
    % Have we reached the maximum for this level?
    \ifnumgreater
      {\value{level\romannumeral\Level}}%
      {\value{maxlevel\romannumeral\Level}}%
    {}% reached maximum, do nothing
    {% Add row number to the appropriate list
      \listcsxadd{listlevel\romannumeral\Level}{\DTLcurrentindex}%
    }%
    % do we need to continue or have we got everything?
    \ifboolexpr
    {%
      test{\ifnumgreater{\value{leveli}}{\value{maxleveli}}}
      and
      test{\ifnumgreater{\value{levelii}}{\value{maxlevelii}}}
    }%
    {\dtlbreak}{}%
  }

\renewcommand{\do}[1]{%
  \dtlgetrow{problems}{#1}%
  \dtlgetentryfromcurrentrow{\Question}{\dtlcolumnindex{problems}{Question}}%
  \dtlgetentryfromcurrentrow{\Answer}{\dtlcolumnindex{problems}{Answer}}%
  \item \Question

   \ifdefempty\Answer
   {}% no answer
   {% do answer if this is the solution sheet
     \iftoggle{showanswers}{Answer: \Answer}{}%
   }%
}

\begin{enumerate}

% do easy questions
\dolistloop{\listleveli}
```

```
% do medium level questions
\dolistloop{\listlevelii}

\end{enumerate}
\end{document}
```

Now, the \DTLforeach loop just stores the row numbers of the required questions in two lists, corresponding to the two different levels. Then each list is iterated through and the corresponding row is fetched using \dtlgetrow. Extending this example to accommodate an arbitrary number of levels is left as an exercise for the reader.

Remember that if you have the shell escape enabled when you run LaTeX you can invoke datatooltk in your document *before* you load the database:

```
\immediate\write18{datatooltk --in data.dbtex --seed 2013 --shuffle
--output data-shuffled.dbtex}

\input{data-shuffled.dbtex}
```

4.3 Sorting and Shuffling

As mentioned earlier, if you specify both --sort and --shuffle, the sorting will always be performed first, regardless of the option order, but why would you want to sort the data if you're going to shuffle it? Consider the command invocation:

```
datatooltk --shuffle --in ⟨in-file⟩ --output ⟨out-file⟩
```

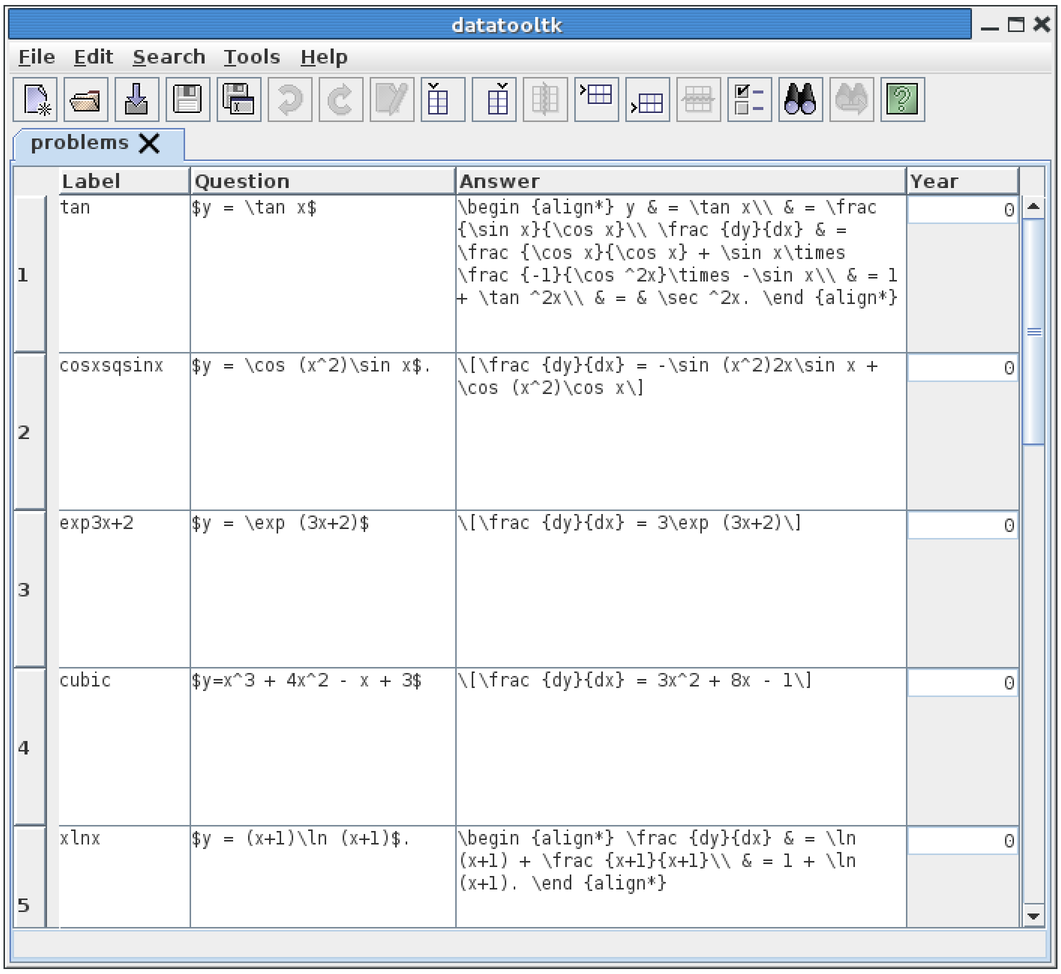
Every time you run this command, you will get a different ordering. If, however, you set a seed for the random generator, for example:

```
datatooltk --seed 2013 --shuffle --in ⟨in-file⟩ --output ⟨out-file⟩
```

You will always get the same random ordering *provided the original data in ⟨in-file⟩ has remained unchanged.* If you want to modify the shuffled data in your document and save it to the original file ⟨*in-file*⟩ using \DTLsaverawdb, the ordering in that file will change, so the next time you shuffle it, you'll get a different ordering, even if you use the same seed. If you sort first on a unique label, that will ensure the shuffle has the same starting point (unless you add or remove rows).

Example 4: Sorting and Shuffling

Suppose you have a database of exam questions and you want to keep track of the year in which each question was last used. (To make life easier, let's identify the academic year "2012/13" as 2013, the academic year "2013/14" as 2014, etc.) Let's further suppose the database of questions is in a file called `mth-101.dbtex` and the database label is "`problems`" (see Figure 4.4). The database contains a column with the label "`Label`", which uniquely identifies an exam question, a column with the label "`Question`" that contains the exam question, a column with the label "`Answer`" that contains the answer and an integer column with the label "`Year`" that contains the exam year in which that question was last used. (A zero entry means the question hasn't been used.)



	Label	Question	Answer	Year
1	tan	$y = \tan x$	\begin {align*} y & = \tan x\\ & = \frac {\sin x}{\cos x}\\ \frac {dy}{dx} & = \frac {\cos x}{\cos x} + \sin x\times \frac {-1}{\cos ^2x}\times -\sin x\\ & = 1 + \tan ^2x\\ & = & \sec ^2x. \end {align*}	0
2	cosxsqsinx	$y = \cos (x^2)\sin x$.	\[\frac {dy}{dx} = -\sin (x^2)2x\sin x + \cos (x^2)\cos x\]	0
3	exp3x+2	$y = \exp (3x+2)$	\[\frac {dy}{dx} = 3\exp (3x+2)\]	0
4	cubic	$y=x^3 + 4x^2 - x + 3$	\[\frac {dy}{dx} = 3x^2 + 8x - 1\]	0
5	xlnx	$y = (x+1)\ln (x+1)$.	\begin {align*} \frac {dy}{dx} & = \ln (x+1) + \frac {x+1}{x+1}\\ & = 1 + \ln (x+1). \end {align*}	0

Figure 4.4: Sort and Shuffle Example

Now suppose the exam requires five questions to be randomly selected from this database, but must not include any question used in the past three years. So the exam LaTeX document needs to load in a shuffled version of `mth-101.dbtex`, use the first five questions that don't have a year set in the past three year range, set the year for the selected questions to the current exam year, display the questions (and optionally the answers for the solution sheet), and at the end of the document, overwrite `mth-101.dbtex` so that it now has a record of this year's exam questions.

There are two problems. Firstly, if the process is to be automated with a call to `datatooltk --shuffle` followed by a LaTeX call, a different set of problems will be selected on each run, even with the same seed. To overcome this, a sort on the "`Label`" column needs to be done before the shuffle:

```
datatooltk --sort Label --seed 2013 --shuffle --in mth-101.dbtex ↩
--output mth-101-shuffled.dbtex
```

(The symbol ↩ above indicates a line wrap. Don't insert a line break at that point.)This way the shuffle always starts from the same ordering.

The second problem occurs if you edit the database such that you add or remove rows. This will change the initial conditions, even with the sort. If you add or remove rows, you need to accept that the document may well end up with a different selection of questions, which is okay if you haven't finalised the exam, but it means that some of the questions will be identified as having been used in that exam year from a previous run but are now no longer selected. In order to make them available for the next year, if they haven't been selected but have had the year set to this year, the year needs to be cleared.

To solve this, once you have selected the maximum required number of questions, don't break out of the loop, as was done earlier (see §4.2). Instead, for the rest of the loop, if the exam year is set to the current year, clear it.

```
% arara: pdflatex: {shell: on}
\documentclass{article}

\usepackage{etoolbox}
\usepackage{datatool}
\usepackage{listings}
%
\newtoggle{showanswers}
\togglefalse{showanswers}

\newcommand{\examyear}{2013}
\newcommand{\maxquestions}{5}
\newcounter{question}

\immediate\write18{datatooltk --sort Label --seed \examyear\space
--shuffle --in mth-101.dbtex --output mth-101-shuffled.dbtex}

\input{mth-101-shuffled.dbtex}
```

```
\begin{document}

\begin{enumerate}
 \DTLforeach{problems}{\Question=Question,\Answer=Answer,\Year=Year}%
 {%
   % If year hasn't been specified, set it to 0 to
   % allow numeric comparisons
   \DTLifnullorempty{\Year}%
   {%
      \def\Year{0}%
      \DTLappendtorow{Year}{0}%
   }%
   {}%
   \ifnumgreater{\value{question}}{\maxquestions}
   {%
      % Finished selecting questions, unset any year
      % equal to this exam year
      \ifnumequal{\Year}{\examyear}
      {%
         % unset year
         \DTLreplaceentryforrow{Year}{0}%
      }%
      {}%
   }%
   {%
      % Still selecting questions.
      % Check the year
      \ifboolexpr
      {%
         test{\ifnumequal{\Year}{\examyear}}
         or
         test{\ifnumless{\Year}{\examyear-3}}
      }
      {% select this question
         \stepcounter{question}%
         \item \Question

         \iftoggle{showanswers}{Answer: \Answer}{}%
         % update year
         \DTLreplaceentryforrow{Year}{\examyear}%
      }%
      {% skip this question, it was used in the past 3 years
      }%
   }%
```

```
  }
\end{enumerate}

% update database file
\DTLprotectedsaverawdb{problems}{mth-101.dbtex}
\end{document}
```

Since this overwrites the datatool file, you will lose any pretty-printing spaces or comments you may have done in datatooltk's cell editor dialog.

4.4 Plugins

Plugins are usually associated with a particular template (see §6) and provide a convenient way of adding a row of data to the currently selected database. Typically when a plugin is run it will add a new row of data if no row is selected, otherwise it will allow you to edit the selected row.

You must have Perl installed to use the plugins (see §7). You will also need to ensure that the Tk module is installed.

For example, on Fedora:

```
sudo dnf install perl-Tk
```

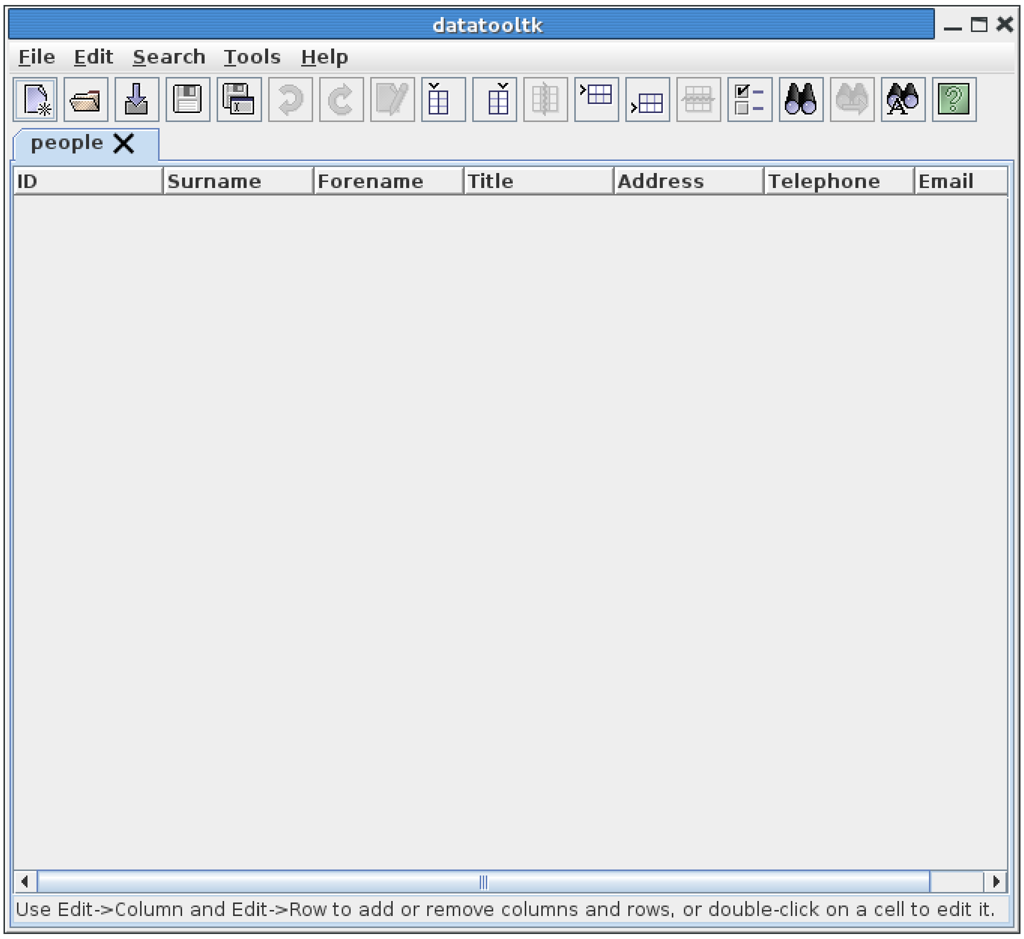
The plugin is sent the database information when you start each instance of the plugin, so if you change the database in datatooltk while a plugin is running there may be unexpected results. Wait until the plugin has finished (usually by clicking on **Okay** or **Cancel**) before you make any further edits to the database.

4.4.1 The people Plugin

The people plugin is designed for use with databases created using the people template.

Example 5: Creating a new Database of People with a Plugin

Suppose you create a new database using the people template. This creates a database with



Use Edit->Column and Edit->Row to add or remove columns and rows, or double-click on a cell to edit it.

Figure 4.5: Database Created From people Template

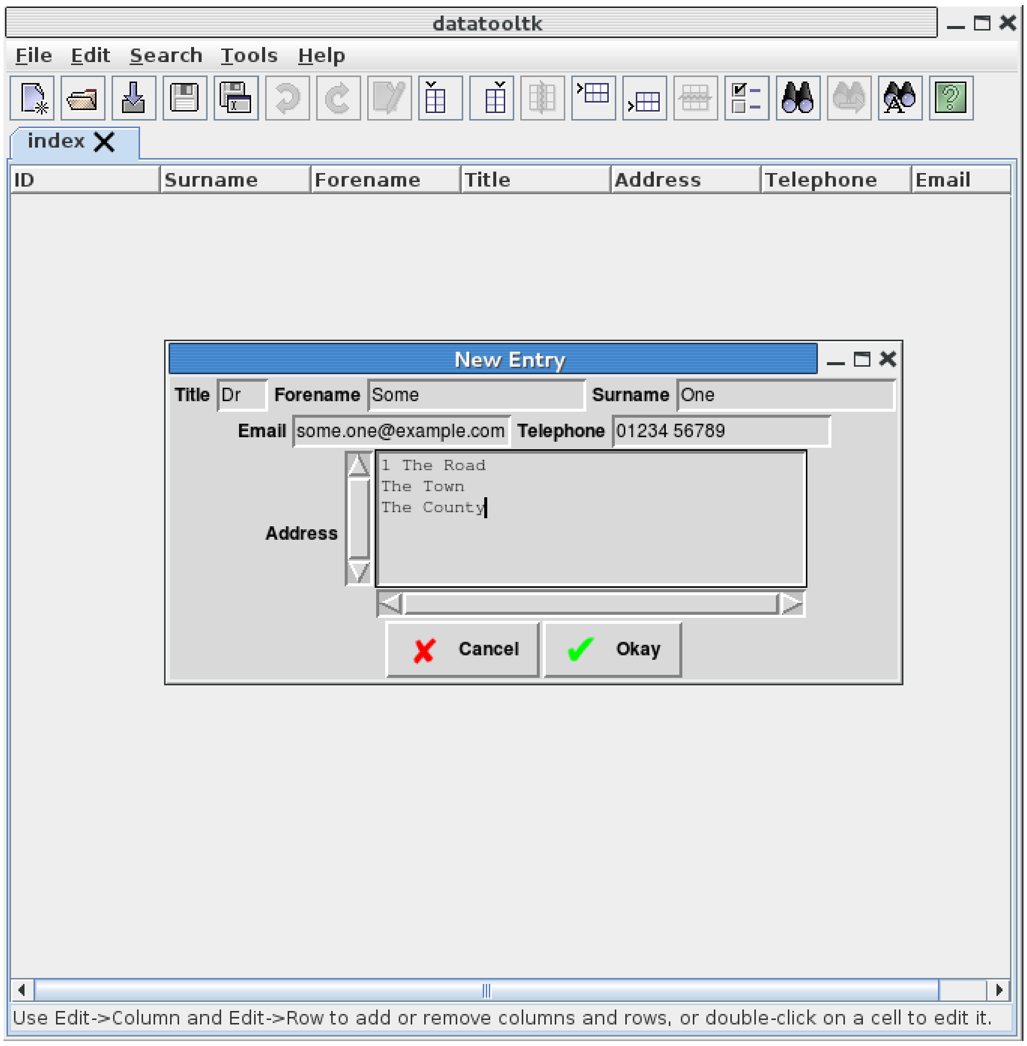
the following fields: ID, Title, Surname, Forename, Address, Telephone and Email, as illustrated in Figure 4.5.

Having created this database, I can just use the **Edit** ➜ **Row** menu to insert rows and then edit each entry, but suppose I want to automatically increment the associated ID for each person. I can do this using the `people` plugin that corresponds to this template via the **Tools** ➜ **Plugins** menu.

If a row is currently selected, this plugin will allow you to edit the data for that row. Otherwise, it will allow you to insert a new row. For a new row of data, the `people` plugin will open the dialog box shown in Figure 4.6.

After entering the data, I can click on **Okay** and a new row of data is added to the database (see Figure 4.7). Note that the plugin has converted newline characters in the address into `\\`. The ID has automatically been inserted.

Since the `people` plugin only adds or modifies a single row at a time, if you no longer require an entry, you can delete the unwanted row using **Edit** ➜ **Row** ➜ **Delete Row**.



Title Dr Forename Some Surname One

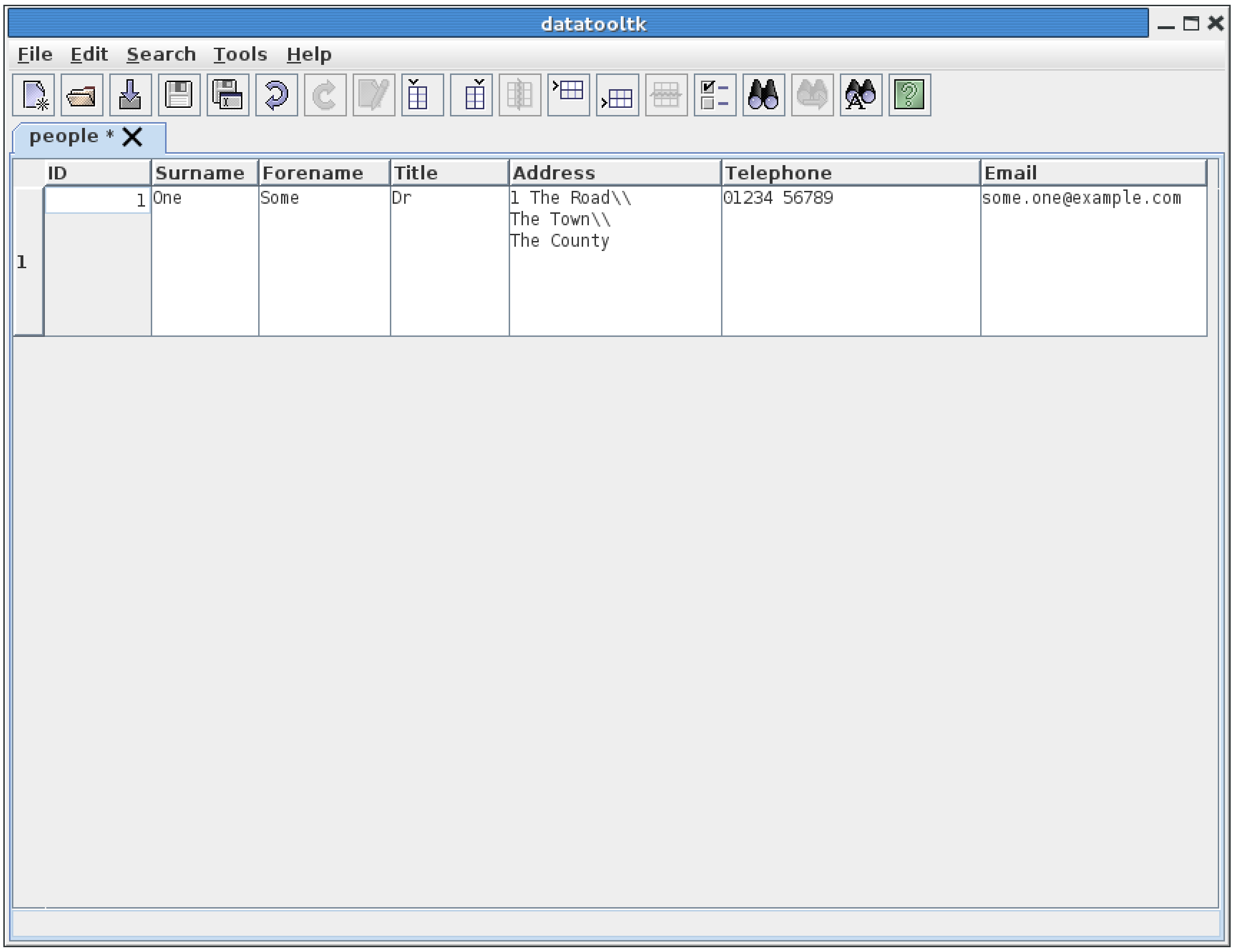
Email some.one@example.com Telephone 01234 56789

Address

```
1 The Road
The Town
The County
```

Use Edit->Column and Edit->Row to add or remove columns and rows, or double-click on a cell to edit it.

Figure 4.6: The people Plugin Dialog



	ID	Surname	Forename	Title	Address	Telephone	Email
1	1	One	Some	Dr	1 The Road\\ The Town\\ The County	01234 56789	some.one@example.com

Figure 4.7: A New Row of Data

4.4.2 The datagidx Plugin

The datagidx package creates its own custom database to store terms, symbols and acronyms. The datagidx template will create a database that contains datagidx's required fields. There are a lot of fields, some of which are reserved for datagidx's private use. The datagidx plugin, available via the **Tools**➜**Plugins** menu, provides a convenient interface to add or edit entries. If no row is selected, the plugin will create a new row. If a row is selected, the plugin will allow you to edit or remove the row. Since the datagidx plugin can modify other rows at the same time (for example, if you set a parent entry or cross-reference) it's recommended that you use the datagidx plugin to remove an entry (via the **Remove Entry** button) rather than using **Edit**➜**Row**➜**Delete Row**.

Example 6: Creating a Database with the datagidx Template

The new database (created via **File**➜**New From Template...**) is shown in Figure 4.8. The default name of the database is "**index**". You can change it as required, but don't call it "datagidx" as the datagidx package creates a database with that name for its private use. Once this database has been created, the datagidx plugin will open the dialog box shown in Figure 4.9.

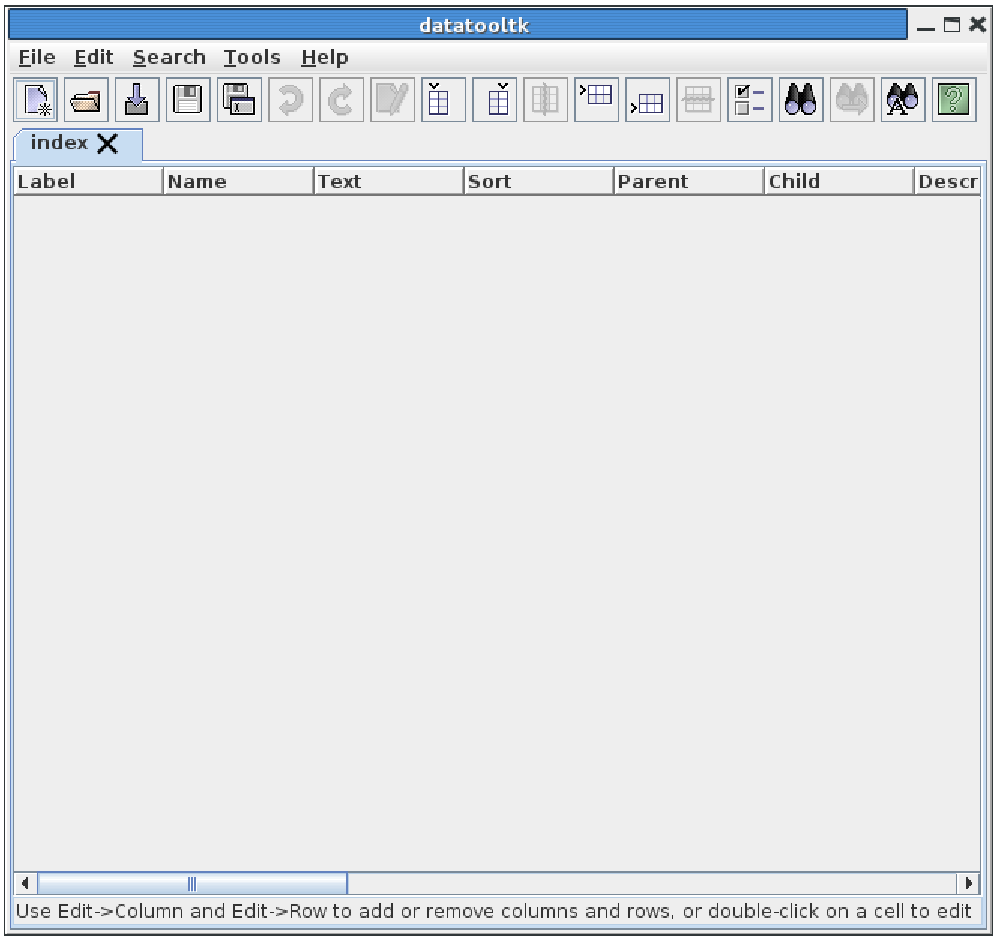
Since many of the fields are often duplicated (for example, the `Name` field is often the same as the `Text` field) if you first enter the name in the **Name** field, when you move the focus to another field, default entries will be added to most of the empty fields. For example, in Figure 4.10 I typed "bird" in the **Name** field and then moved the cursor to the **Description** field. This automatically filled in default values for the **Label**, **Sort**, **Text**, **Short**, **Long**, **Plural**, **Short Plural** and **Long Plural** fields. Since this is the first entry, there are no options for the **Parent**, **See** and **See Also** fields. (The last two are hidden in Figure 4.10 as the **Cross-Reference** button is unchecked.)

When I click on **Okay**, a new row is added to the database (see Figure 4.11). Note that I didn't specify a parent for this entry so the parent has been given the value `\@dtlnovalue`, which ensures it will work correctly when the datagidx package tests if the parent entry is null.

If I use the datagidx plugin to create a new row, there are now options available in the **Parent**, **See** and **See Also** fields (see Figure 4.12).

In Figure 4.12 I have set the parent to `bird`. When the new row is added, the plugin automatically adjusts the `bird` entry to include the new `duck` label as one of its children (see Figure 4.13).

It's also possible to cross-reference another entry. There are two ways of cross-referencing an entry: (1) using **See** which redirects the reader to a synonym that has the location list; (2) using **See Also** which in addition to the location list refers the reader to one or more related topics. (See the datagidx section of the datatool user manual for further details.) To enable either form of cross-referencing, make sure the **Cross-Reference** button is selected. This will display extra options, shown in Figure 4.14.



Use Edit->Column and Edit->Row to add or remove columns and rows, or double-click on a cell to edit

Figure 4.8: A Database Created From the datagidx Template

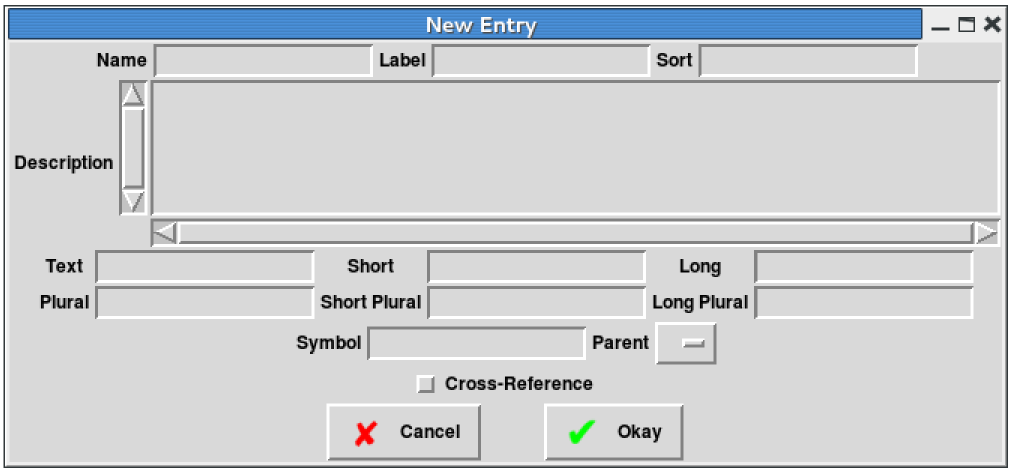
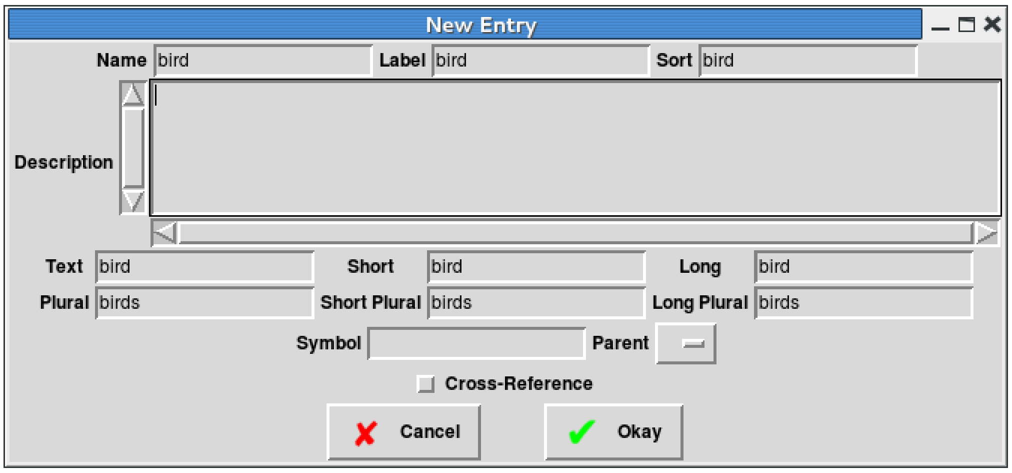


Figure 4.9: The datagidx Plugin Dialog



Name	bird	Label	bird	Sort	bird
Text	bird	Short	bird	Long	bird
Plural	birds	Short Plural	birds	Long Plural	birds
Symbol		Parent			

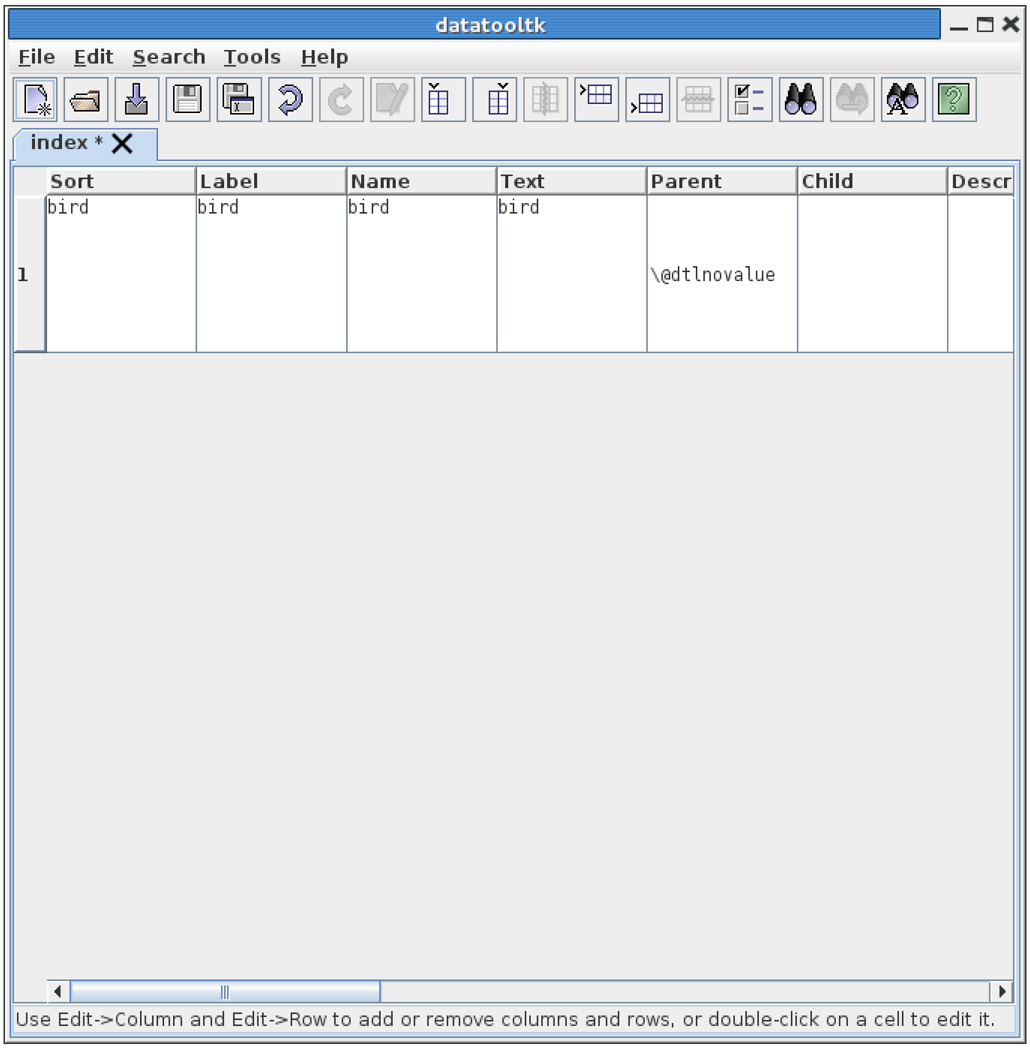
Figure 4.10: Most Fields Are Auto-Filled From the Name Field

Either select the **See** button and choose the synonym from the drop-down box next to it, or select the **See Also** button and select the related cross-reference from the drop-down box to the right and either click on **Add "See Also" Entry** to append it to the **See Also** list or click on **Remove "See Also" Entry** to remove it from the list. For example, in Figure 4.14 I've added the `chicken` and `turkey` entries to the **See Also** list. (Assuming I've already added the `chicken` and `turkey` entries before defining this new entry.)

Once I've enter all my terms, I can sort the data according to the **Sort** column. (Recall §4.1.) Now let's suppose I save this sorted database to a file called `datagidx-test.dbtex`. I can now load it into a LaTeX document as follows:

```
\documentclass{article}

\usepackage{datagidx}

\loadgidx{datagidx-test.dbtex}{Index}

\begin{document}

Reference some terms: \gls{duck}, \gls{bird}, \gls{parrot},
\gls{crocodile}, \gls{caiman}, \gls{alligator}.

\printterms[columns=1]

\end{document}
```

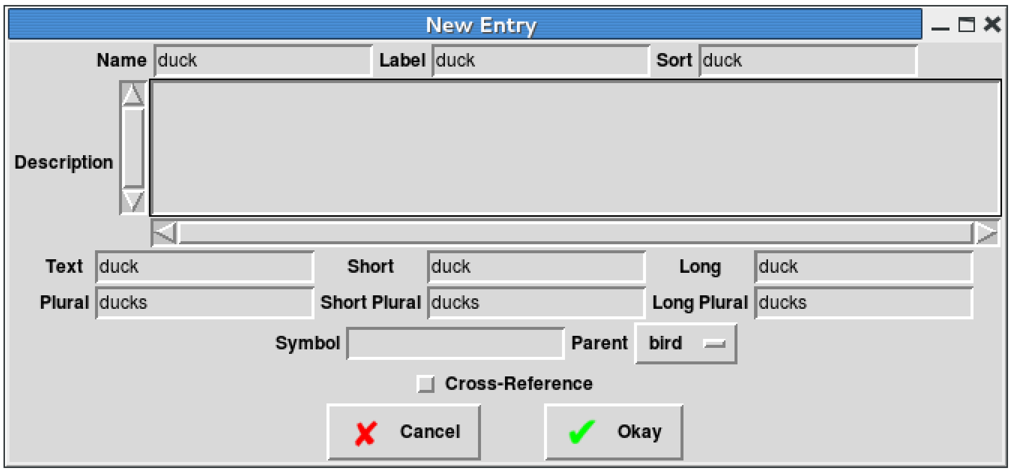
Ensure you have at least version 2.15 of the `datatool` bundle.



	Sort	Label	Name	Text	Parent	Child	Descr
1	bird	bird	bird	bird	\@dtlnovalue		

Use Edit->Column and Edit->Row to add or remove columns and rows, or double-click on a cell to edit it.

Figure 4.11: New Row Added to the Database


New Entry

Name duck Label duck Sort duck

Description

Text duck Short duck Long duck

Plural ducks Short Plural ducks Long Plural ducks

Symbol Parent bird

Cross-Reference

Figure 4.12: Parent Field Lists Other Entry Labels

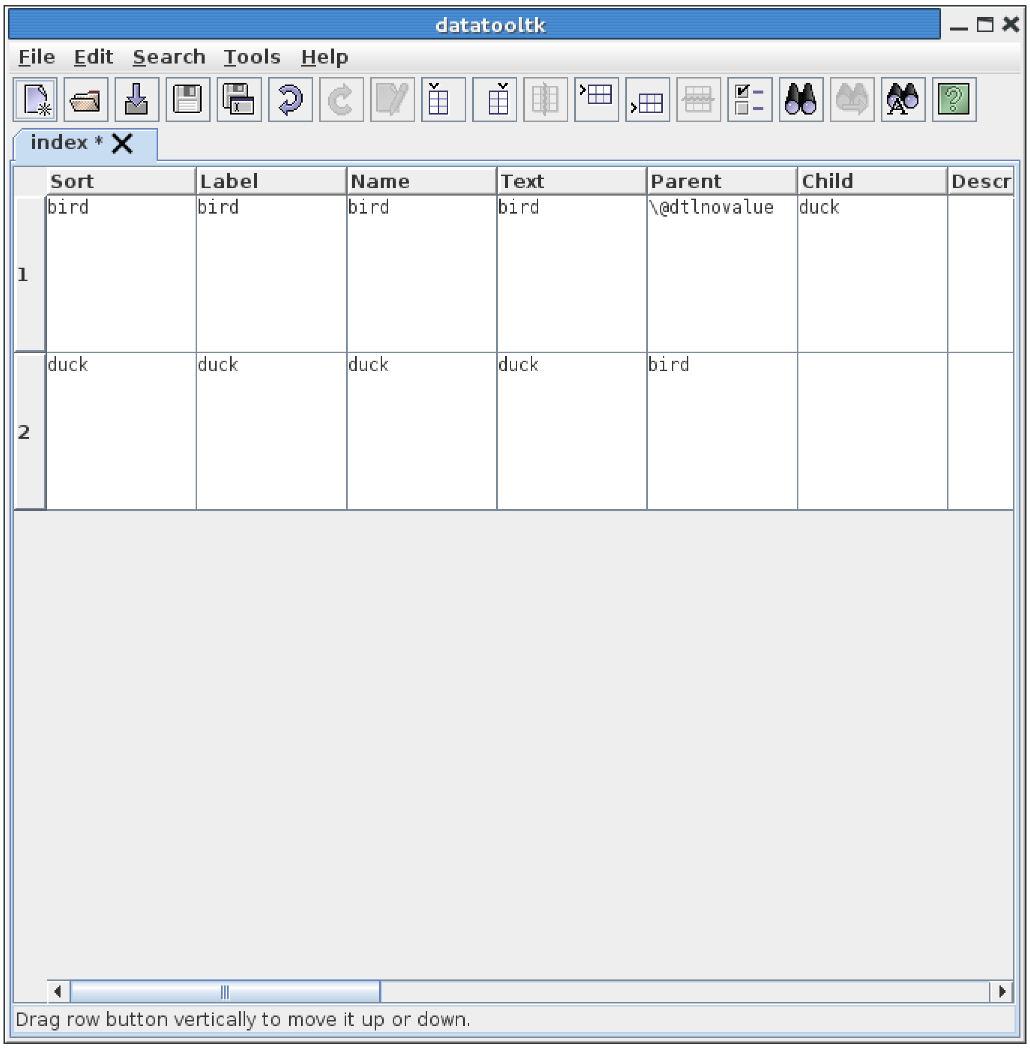
4.4.3 Comparison of glossaries and datagidx

If you're interested in the comparative efficiency between using glossaries and datagidx, I performed a test with 100 entries randomly selected from a dictionary. The entries were listed in a file called entries in the form:

```
\newterm{minnow}
\newterm{running board}
\newterm{diamanté}
```

First, let's look at a document that uses datagidx with \newgidx:

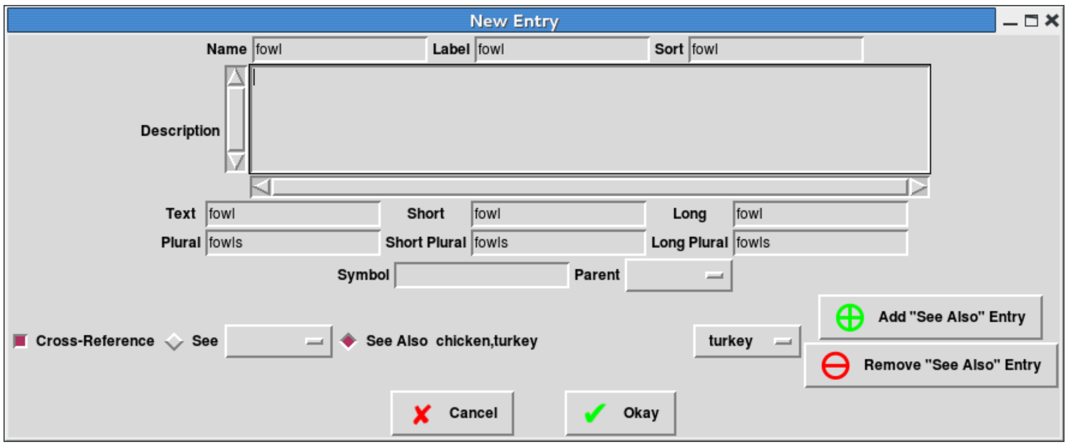
```
% arara: clean: {extensions: ['aux']}
% arara: pdflatex
% arara: pdflatex
\documentclass{report}

\usepackage{datagidx}

\newgidx{index}{Index}

\input{entries}

\begin{document}

\tableofcontents
```



	Sort	Label	Name	Text	Parent	Child	Descr
1	bird	bird	bird	bird	\@dtlnovalue	duck	
2	duck	duck	duck	duck	bird		

Drag row button vertically to move it up or down.

Figure 4.13: Child Entry Automatically Adjusted For Parent Entry


New Entry

Name: fowl | Label: fowl | Sort: fowl

Description:

Text: fowl | Short: fowl | Long: fowl

Plural: fowls | Short Plural: fowls | Long Plural: fowls

Symbol: | Parent:

Cross-Reference ◇ See ◆ See Also chicken,turkey turkey

Figure 4.14: Cross-Referencing Entries

```
\chapter{Sample}
\glsaddall{index}

\printterms[postheading={\addcontentsline{toc}{chapter}{Index}}]

\end{document}
```

In general you need three LaTeX runs to compile a datagidx document. In this case, you actually only need to do it twice since there are no location lists.

Now let's test a datagidx document where datatooltk does the sorting. First, we need to generate a dbtex file that corresponds the same set of entries. This can be done with the following document:

```
\documentclass{article}

\usepackage{datagidx}

\newgidx{index}{Index}

\input{entries}

\begin{document}

\DTLprotectedsaverawdb{index}{index.dbtex}

\mbox{}\newpage

\end{document}
```

This just converts the entries listed in entries.tex into the appropriate database file, simulating having entered the terms using datatooltk's datagidx plugin. The file is saved as index.dbtex. Remember that this data only needs to be sorted when you add a term. This can either be done in datatooltk's GUI mode or it can be done in batch mode:

```
datatooltk --in index.dbtex --sort Sort --output index-sorted.dbtex
```

This creates a file called index-sorted.dbtex. This file can be loaded into a document as follows:

```
% arara: clean: {extensions: ['aux']}
% arara: pdflatex
% arara: pdflatex
\documentclass{report}

\usepackage{datagidx}

\loadgidx{index-sorted.dbtex}{Index}

\begin{document}

\tableofcontents

\chapter{Sample}

\glsaddall{index}

\printterms[postheading={\addcontentsline{toc}{chapter}{Index}}]

\end{document}
```

Now let's look at the glossaries package. Since the terms have been defined using \newterm, I've defined a command that will convert this into an equivalent \newglossaryentry. Some of the entries have accents in their name, which datagidx automatically strips when generating the default label, so I've added a quick way of generating an analogous accent-free label and sort key that can be used with \newglossaryentry. Here's the document:

```
% arara: clean: {extensions: ['aux', 'gls']}
% arara: pdflatex
% arara: makeglossaries
% arara: pdflatex
\documentclass{report}

\usepackage[nonumberlist,nogroupskip,toc]{glossaries}
```

```
\usepackage{glossary-mcols}

\makeglossaries
\renewcommand{\glossaryname}{Index}
\renewcommand{\glsnamefont}[1]{\textmd{#1}}

\newcommand{\newterm}[1]{%
  \bgroup
    \def\c##1{##1}%
    \let\'\c
    \xdef\thislabel{#1}%
  \egroup
  \def\thisname{#1}%
  \edef\donewgloss{%
    \noexpand\newglossaryentry{\thislabel}%
    {name={\expandonce\thisname},%
     sort={\thislabel},%
     description={\noexpand\nopostdesc}}%
  }%
  \donewgloss
}

\input{entries}

\begin{document}
\tableofcontents

\chapter{Sample}
\glsaddall

\printglossary[style=mcolindex]

\end{document}
```

In order to compare them, I used `arara` with the Linux `time` command. In each case, the `clean` directive is used at the start to ensure the tests start without any auxiliary files. Since there are no location lists, only two LaTeX calls are used on each example. If there were location lists, the `datagidx` examples would both need a third LaTeX call.

The `datagidx` package doesn't generate a location with `\glsadd` or `\glsaddall`, whereas `glossaries` does. I've suppressed the location list in the `glossaries` example to produce an equivalent document.

Remember that with the example that uses index-sorted.dbtex, datatooltk needs to sort the database whenever a new entry is added to the database. Assuming that all possible required entries have been added to the database, we just need one sort operation:

```
datatooltk --in index.dbtex --sort Sort --output index-sorted.dbtex
```

Invoking this with the Linux time command gives:

```
real 0m0.296s
user 0m0.466s
sys 0m0.033s
```

Now arara can be run on each of the three test documents (via the time command). The result from the first test that uses datagidx and \newgidx. The results are:

```
real 0m13.801s
user 0m13.925s
sys 0m0.076s
```

arara records the total time taken as 13.39 seconds.

The next test uses datagidx and \loadgidx. The result is:

```
real 0m2.643s
user 0m2.775s
sys 0m0.065s
```

arara records the total time taken as 2.23 seconds.

The third test uses glossaries. The result is:

```
real 0m1.156s
user 0m1.307s
sys 0m0.069s
```

arara records the total time taken as 0.75 seconds.

Using glossaries is clearly faster than using datagidx. In the case of \loadgidx, glossaries is approximately three times faster. In the case of \newgidx, glossaries is approximately 18 times faster. If a third LaTeX run was required for the location lists with \newgidx, using glossaries would be approximately 27 times faster (with only two LaTeX runs and one makeglossaries run).

If you're interested to know how this compares with \makenoidxglossaries instead of \makeglossaries, here's the revised glossaries code:

```
% arara: clean: {extensions: ['aux']}
% arara: pdflatex
% arara: pdflatex
```

```
\documentclass{report}

\usepackage[nonumberlist,nogroupskip,toc]{glossaries}
\usepackage{glossary-mcols}

\makenoidxglossaries

\renewcommand{\glossaryname}{Index}
\renewcommand{\glsnamefont}[1]{\textmd{#1}}

\newcommand{\newterm}[1]{%
  \bgroup
    \def\c##1{##1}%
    \let\'\c
    \xdef\thislabel{#1}%
  \egroup
  \def\thisname{#1}%
  \edef\donewgloss{%
    \noexpand\newglossaryentry{\thislabel}%
    {name={\expandonce\thisname},%
     sort={\thislabel},%
     description={\noexpand\nopostdesc}}%
  }%
  \donewgloss
}

\input{entries}

\begin{document}
\tableofcontents

\chapter{Sample}
\glsaddall

\printnoidxglossary[style=mcolindex]

\end{document}
```

The result is:

```
real 0m2.463s
user 0m2.596s
sys 0m0.065s
```

arara records the total time taken as 2.06 seconds. This is slightly quicker than the second datagidx test.

5 Importing Data

Data can be imported from CSV files (see §5.1), SQL databases (see §5.3) or from files that can be imported with the `probsoln` package's `\loadallproblems` command (see §5.4). In the case of the first two, DatatoolTk can automatically convert TeX's special characters if the `--map-tex-specials` (or `--literal`) command line option is used or the **Literal** option has been selected in the **Preferences** dialog box (see §7). Both the column title and label will be obtained from the appropriate data header. The title will have any mappings applied (if set). The label will have forbidden content (control sequences and the standard set of special characters) removed. In the case of CSV files or spreadsheets imported without headers, default values will be used.

Note that data can't be exported back to any of those formats.

5.1 Import CSV Data

Data can be imported from a CSV file using the `--csv` command line option or (in GUI mode) using the ???? menu item. The default separator is a comma (`,`) and the default delimiter is the double-quote (`"`) character. These can be changed using the `--csv-sep` and `--csv-delim` command line options or in the **Preferences** dialog box (see §7).

If the CSV file has a header row, you must make sure the `--csvheader` option is used or the **Has Header Row** option is checked in the **Preferences** dialog box. If the CSV file has no header row, you must make sure the `--nocsvheader` option is used or the **Has Header Row** option is unchecked in the **Preferences** dialog box.

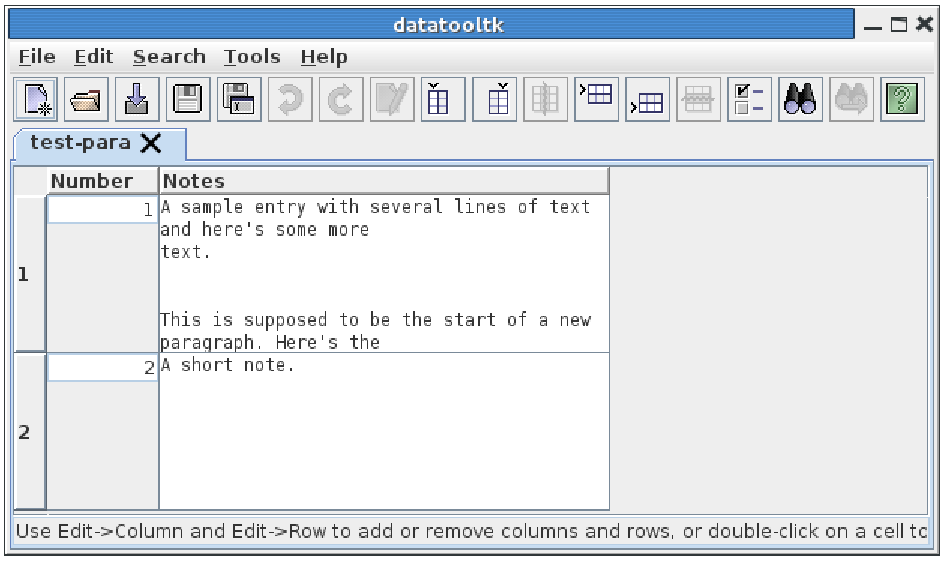
Make sure that the CSV file encoding is correctly set before importing. This can be done from the `--csv-encoding` command line option or in the **Preferences** dialog box. The encoding of the TeX (`dbtex`) file is independent of the CSV encoding.

Example 7: Importing Data From a CSV File

Consider the CSV file shown below:

```
Number,Notes
1,"A sample entry with several lines of text and here's some more
text.

This is supposed to be the start of a new paragraph. Here's the
next sentence."
2,A short note.
```

This has a cell with multiple lines. When it's imported into `datatooltk`, the paragraph break



	Number	Notes
1	1	A sample entry with several lines of text and here's some more text. This is supposed to be the start of a new paragraph. Here's the
2	2	A short note.

Use Edit->Column and Edit->Row to add or remove columns and rows, or double-click on a cell tc

Figure 5.1: Paragraph Breaks Appear as a Single Blank Line

is converted to \DTLpar. However, this isn't visible when you look at the file in GUI mode (see Figure 5.1).

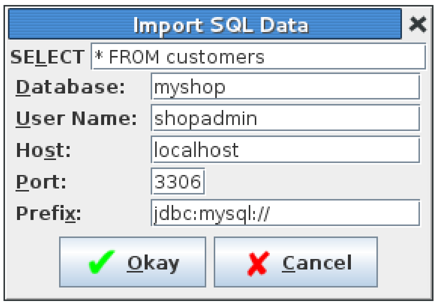
Note that the redundant second blank line in the CSV file has gone as multiple blank lines are replaced by a single \DTLpar.

5.2 Import Spreadsheet Data

Data can be imported from an Excel xls file (via the Apache POI library http://poi.apache.org/) or an Open Document Spreadsheet using the --xls or --ods command line options, respectively. Alternative, in GUI mode you can use the ???? menu item. Note that the xlsx is currently unsupported.

When using the command line, you additionally need to specify the sheet index (starting from 0) or the sheet name using the --sheet command line option. If you are using the GUI, after you've selected the spreadsheet file from the file selector dialog, you need to select the required sheet name. Importing data from a spreadsheet uses the same header row option and TeX mapping settings as the import CSV function. So if the sheet doesn't have a header row, you need to use --nocsvheader or uncheck the **Has Header Row** button in the **Preferences** dialog box. Note that xlsx files aren't supported.

No formatting information is read when importing data from an Excel spreadsheet. It's up to you to explicitly add LaTeX font commands when you include the data in your document.


Import SQL Data

SELECT * FROM customers

Database: myshop

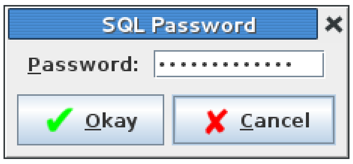
User Name: shopadmin

Host: localhost

Port: 3306

Prefix: jdbc:mysql://

Figure 5.2: SQL Import Dialog Box


SQL Password

Password: •••••••••••••

Figure 5.3: Password Dialog Box

5.3 Import SQL Data

Data can be imported from an SQL database using the `--sql` command line option or the ???? menu item. You additionally need to supply the database, port, prefix, host, user name and password. In batch mode, you can use the command line options `--sqldb`, `--sqlport`, `--sqlprefix`, `--sqlhost` and `--sqluser`. Although you can specify the password with `--sqlpassword`, this isn't recommended as it isn't secure. If you don't use `--sqlpassword`, you will be prompted for the password, where the text you enter won't be visible. Note that in batch mode, the default action is to use the console to request the password. If there's no console available, you need to use the `--noconsole-action` to determine what action to perform. See §1 for more details about command line options.

In GUI mode, when you use ???? the dialog box shown in Figure 5.2 will be displayed, where you can enter the settings. In addition to the above named settings, you must also specify the SQL SELECT statement that identifies the required data to import. (This manual assumes that if you have data in an SQL database, then you have a basic knowledge of SQL syntax.)

For example, in Figure 5.2 I want to import all data from the table called "customers" in the MySQL database called "myshop". (I've created a user called "shopadmin" with SELECT privileges for this database.) Once I've entered this information, I then click on **Okay** and the password dialog box will appear (see Figure 5.3).

Alternatively, I can use batch mode to import and save the data from the command prompt:

```
datatooltk --output customers.dbtex --sql "SELECT * FROM customers"
↩
--sqldb myshop --sqluser shopadmin
Password:
```

(The symbol ↩ above indicates a line wrap. Don't insert a line break at that point.)The password should be entered at the **Password:** prompt. Remember that it's more efficient to get the SQL database to do any sorting. For example (assuming the table has a column called `Surname`):

```
datatooltk --output customers.dbtex --sql "SELECT * FROM customers
ORDER BY
↩
Surname" --sqldb myshop --sqluser shopadmin
Password:
```

5.4 Import probsoln Data

The `probsoln` package allows you to define problems (and optionally their solutions) using `\newproblem` or the defproblem environment. `datatooltk` can load a file containing these definitions and convert the `probsoln` data into a `datatool` database containing three or four columns with keys: **Label**, **Question** and **Answer**. If the imported data contains multiple `probsoln` data sets, the fourth column has the key **Data Set** and contains the dataset label. You can import one of these files using the `--probsoln` command line option or (in GUI mode) using the ???? menu item. If you have a large number of problems, you can speed things up by setting the initial capacity to that number. (If the initial capacity is smaller than the total number of problems, the hash map used to store the data will have to be enlarged whenever the current capacity is exceeded.)

TEX is a difficult language to parse, so `datatooltk` uses the TEX Parser Library to help gather the data from the imported file. Earlier versions of `datatooltk` used LATEX but this is no longer required. This import function is governed by the TEX file encoding and the **Strip solution environment from probsoln problems** settings (see §7).

If any problems require arguments the default values will be used.

Example 8: Importing Data From a probsoln File

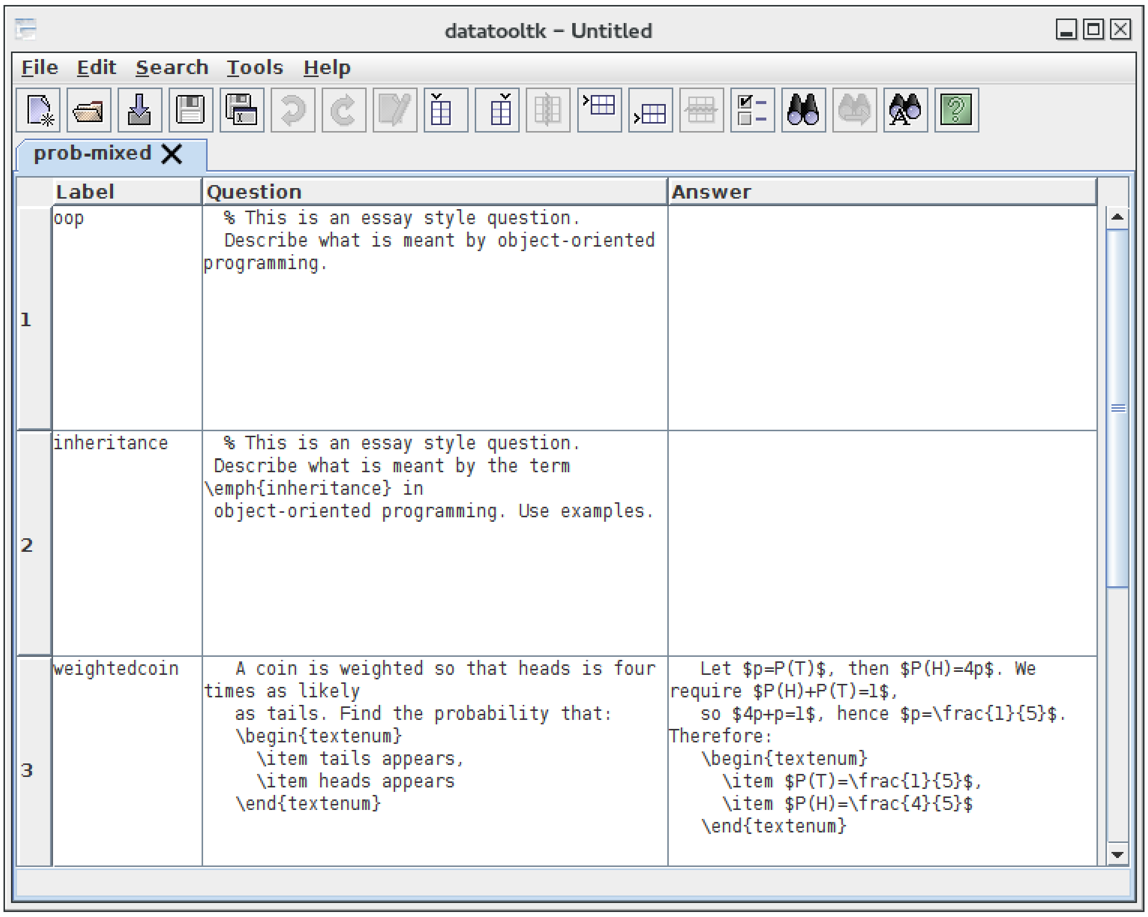
Consider the file called `prob-mixed.tex` that contains the following:

```
\newproblem*{oop}{%
  % This is an essay style question.
  Describe what is meant by object-oriented programming.%
}

\begin{defproblem}{inheritance}
  % This is an essay style question.
 Describe what is meant by the term \emph{inheritance} in
 object-oriented programming. Use examples.
\end{defproblem}

\newproblem{weightedcoin}%
{%
   A coin is weighted so that heads is four times as likely
   as tails. Find the probability that:
   \begin{textenum}
     \item tails appears,
     \item heads appears
   \end{textenum}%
}%
{%
   Let $p=P(T)$, then $P(H)=4p$. We require $P(H)+P(T)=1$,
   so $4p+p=1$, hence $p=\frac{1}{5}$. Therefore:
   \begin{textenum}
     \item $P(T)=\frac{1}{5}$,
     \item $P(H)=\frac{4}{5}$
   \end{textenum}
}

\begin{defproblem}{validprobspaces}
\begin{onlyproblem}%
Under which of the following functions does
$S=\{a_1,a_2\}$ become a probability space?
\par
\begin{textenum}
\begin{tabular}{ll}
\item $P(a_1)=\frac{1}{3}$, $P(a_2)=\frac{1}{2}$
&
\item\label{validprobspacescorrect1} $P(a_1)=\frac{3}{4}$,
$P(a_2)=\frac{1}{4}$
\\
\item\label{validprobspacescorrect2} $P(a_1)=1$, $P(a_2)=0$
&
\item $P(a_1)=\frac{5}{4}$, $P(a_2)=-\frac{1}{4}$
```

```
\end{tabular}
\end{textenum}
\end{onlyproblem}%
\begin{onlysolution}%
\ref{validprobspacescorrect1} and \ref{validprobspacescorrect2}%
\end{onlysolution}
\end{defproblem}

\begin{defproblem}{digraph}
  % This problem requires the tikz package
  \begin{onlyproblem}\label{ex:digraph}
  Identify, if any, the sinks and sources of the digraph shown
  in Figure~\ref{fig:digraph}.

  \begin{figure}[tbh]
    \centering
      \begin{tikzpicture}[every node/.style={draw,circle}]
         \path (0,0) node (A) {$A$}
               (1,0) node (B) {$B$}
               (0,1) node (C) {$C$};
         \draw[->] (A) -- (B);
         \draw[->] (B) -- (C);
         \draw[->] (A) -- (C);
      \end{tikzpicture}
    \par
    \caption{Digraph for Question~\ref{ex:digraph}}
    \label{fig:digraph}
  \end{figure}
  \end{onlyproblem}
  \begin{onlysolution}
  $A$ is a source and $C$ is a sink.
  \end{onlysolution}
\end{defproblem}
```

This contains a mixture of `\newproblem` and defproblem. It also has comments and spaces to make the code more readable. As can be seen in Figure 5.4 these are now still in the import (whereas in older versions they were lost).

The problem defined with the unstarred version of `\newproblem` has a different result depending on whether or not the **Strip solution environment from probsoln problems** setting is on. The normal definition of this command (as provided by `probsoln`) wraps the solution (given in the final argument) in the solution environment. This is stripped when the setting is on, otherwise it's included in the “ `Answer` ” column.



	Label	Question	Answer
1	oop	% This is an essay style question. Describe what is meant by object-oriented programming.	
2	inheritance	% This is an essay style question. Describe what is meant by the term \emph{inheritance} in object-oriented programming. Use examples.	
3	weightedcoin	A coin is weighted so that heads is four times as likely as tails. Find the probability that: \begin{textenum} \item tails appears, \item heads appears \end{textenum}	Let $p=P(T)$, then $P(H)=4p$. We require $P(H)+P(T)=1$, so $4p+p=1$, hence $p=\frac{1}{5}$. Therefore: \begin{textenum} \item $P(T)=\frac{1}{5}$, \item $P(H)=\frac{4}{5}$ \end{textenum}

Figure 5.4: Pretty Printing and Comments are No Longer Lost When Importing Data from probsoln

see also:

- 4.2 Shuffling the Data
- 4.3 Sorting and Shuffling

6 Templates

Templates that come with datatooltk are located in the resources/templates subdirectory of the datatooltk installation directory. You can also write your own templates and store them in the user templates directory (see §6.1). Each template defines a set of column headers. To create a new database with a particular set of column headers, use the **File ➜ New From Template…** menu item, which opens the dialog box shown in Figure 6.1.

The datatooltk application comes with the following templates: datagidx (creates a database with the same structure as used by the datagidx package) and people (creates a database suitable for storing records about people, including columns for forenames, a surname, title and address.) For example, Figure 6.2 shows a database created from the people template.

Rows can now be added to this database using the **Edit ➜ Row** menu or via corresponding plugins (see §4.4).

6.1 Writing a Template File

If you want to write your own template, you need to create an XML file and store it in a subdirectory of the datatooltk user properties directory (see §7) called templates. You will need to create this directory, if it doesn't already exist. For example, on a UNIX-like system, the user template directory will be ~/.datatooltk/templates/. The template file must have the extension xml for it to be listed in the **New From Template** dialog box. (The base name of the file is used in the list.)

The template file must have one <datatooltktemplate> element. This element may contain one or more <header> elements. Each <header> element must contain one <label> element and optionally one <title> and/or one <type> element.

The <label> element contains the uniquely identifying header label. The <title> element contains the header title. If omitted, the title is set to the label, unless there is an entry in the resource dictionary file that matches plugin.⟨*template name*⟩.⟨*label*⟩, in which case that

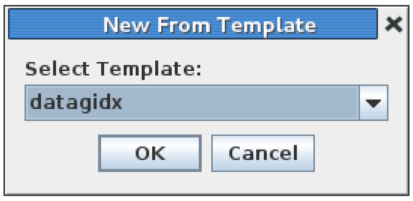
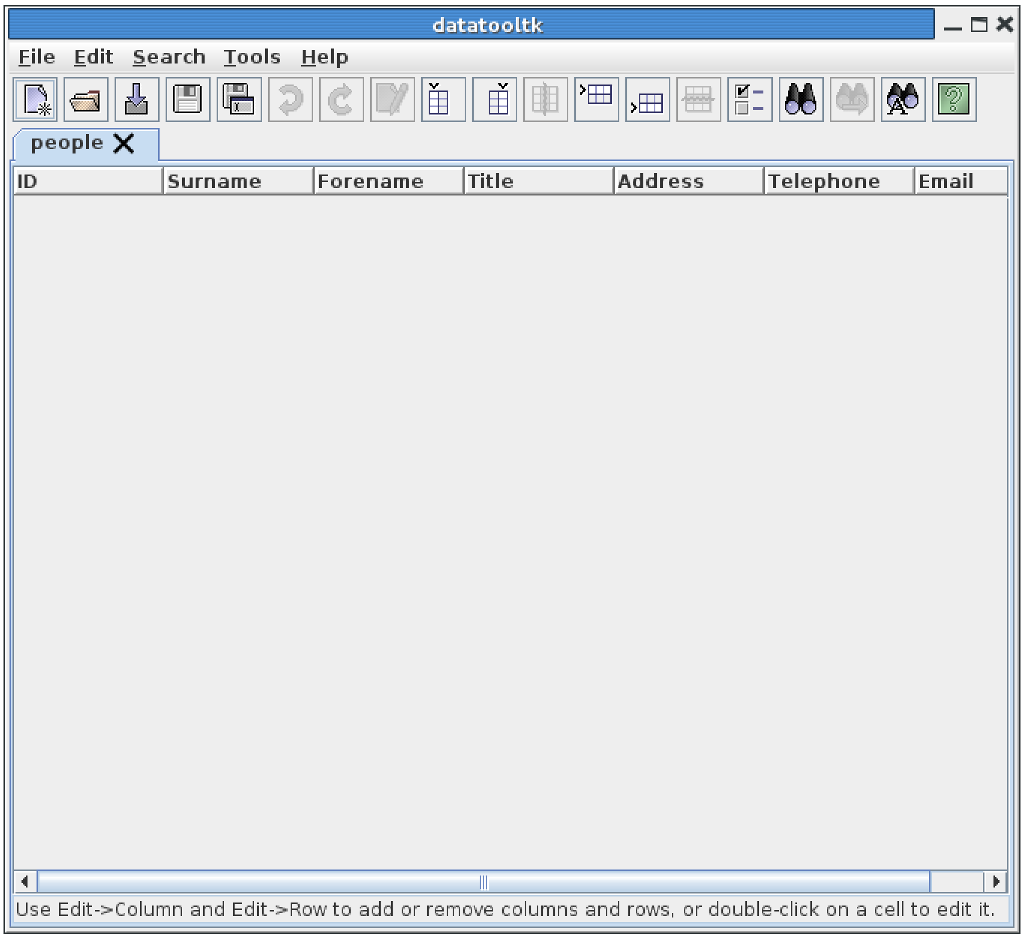


Figure 6.1: New From Template Dialog



Use Edit->Column and Edit->Row to add or remove columns and rows, or double-click on a cell to edit it.

Figure 6.2: New Database Created from people Template

property is used. The <type> element must be one of: -1 (unknown type), 0 (string type),1 (integer type), 2 (real type) or 3 (currency type). If omitted the type is set to -1.

Example 9: Creating a Products Template

Suppose I want to write a template to create a database for a list of products. The database needs three columns: one for the product name, one for the product code and one for the product price. The name should be a string, the price column could either be set to "real" if you don't need to worry about the currency unit or "currency" if you need a currency unit for each product. Let's suppose that the code must be an integer. Here's a template file (the price column is set to "real" rather than "currency"):

```
<datatooltktemplate>
  <header>
    <label>Name</label>
    <type>0</type>
  </header>
  <header>
    <label>Code</label>
    <type>1</type>
  </header>
  <header>
    <label>Price</label>
    <type>2</type>
  </header>
</datatooltktemplate>
```

7 Application Properties

When `datatooltk` is run, either in batch or GUI mode, the application settings are read in from the user properties file, if it exists. Any command line options override those settings. If `datatooltk` is run in GUI mode, the application properties are saved on exit. They are not saved in batch mode, so if you want to change the default settings for batch mode, without having to use the applicable command line option, you will need to run `datatooltk` in GUI mode to set them as required.

The user properties directory depends on the operating system. On Windows, it is a folder called datatooltk-settings in the folder given by the Java system property `user.home`. This is usually the user's home folder but in some versions of Java this can be `%userprofile%`. On other operating systems, the user properties directory is called `.datatooltk` and is in the user's home directory. Alternatively, set the environment variable DATATOOLTK to the directory of your choice.

In GUI mode, the settings can be changed using **Edit ➜ Preferences...**. This opens the **Preferences** dialog box, which has the following tabs.

7.1 General

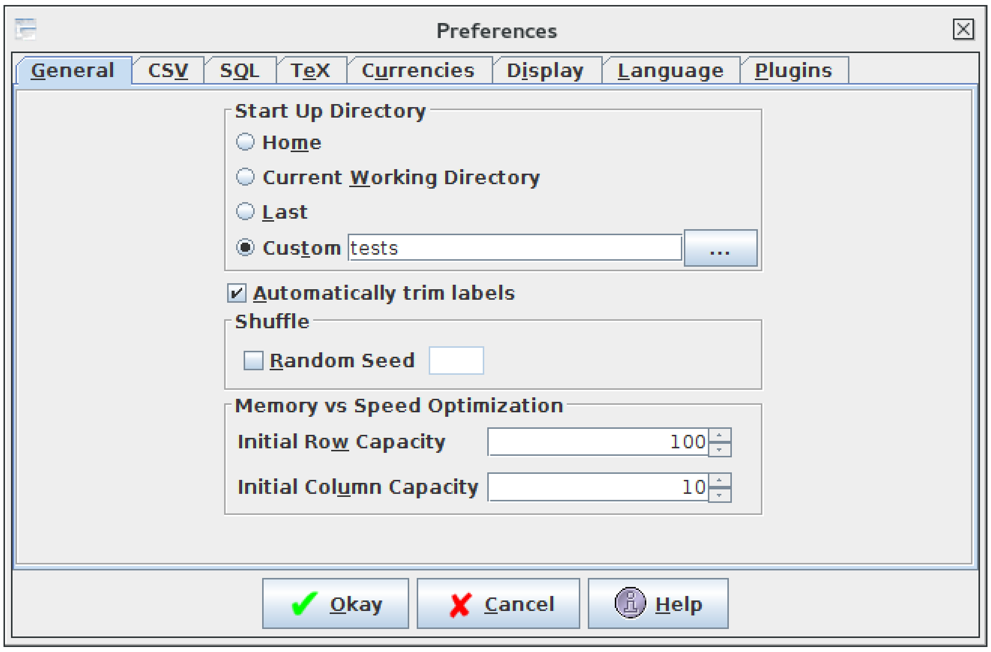
In the **General** tab (Figure 7.1) you can specify the start up directory. (The default directory when you first load, save or import data via the **File** menu.) You can set this to your home directory, the current working directory, the directory you last used on the previous run of `datatooltk` or you can specify a directory of your choice.

In this tab you can also specify the seed for the random number generator (equivalent to `--seed`) and whether or not to automatically strip leading and trailing spaces from database and column labels (equivalent to `--auto-trim-labels` and `--noauto-trim-labels`).

The initial capacity can be increased to speed up loading or importing. Ideally it's best to keep it around the typical size of your databases. If it's too big you can run out of memory. If it's too small, the storage has to be enlarged every time the current capacity is exceeded. The minimum allowed value is 10.

7.2 Non-TeX Imports (CSV etc)

In the **Non-TeX Imports (CSV etc)** tab (Figure 7.2) you can specify the CSV settings. Some of these settings are also used by the spreadsheet import functions.



Start Up Directory

○ Home

○ Current Working Directory

○ Last

● Custom tests

☑ Automatically trim labels

Shuffle

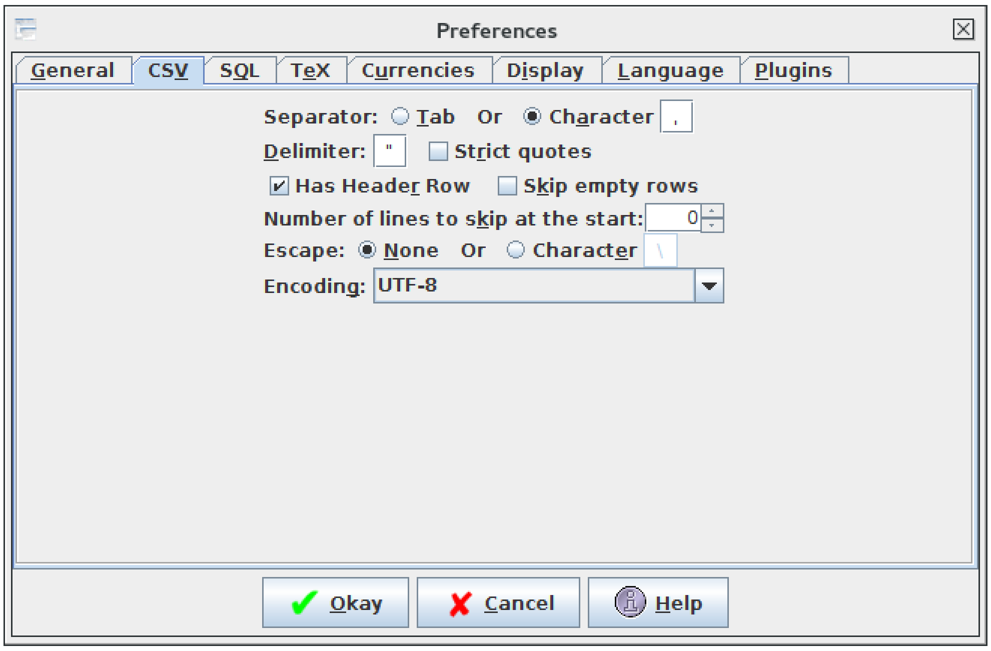
☐ Random Seed

Memory vs Speed Optimization

Initial Row Capacity 100

Initial Column Capacity 10

Figure 7.1: General Tab



Separator: ○ Tab Or ● Character ,

Delimiter: " ☐ Strict quotes

☑ Has Header Row ☐ Skip empty rows

Number of lines to skip at the start: 0

Escape: ● None Or ○ Character \

Encoding: UTF-8

Figure 7.2: CSV Tab

If the separator is a tab character, select the **Tab** radio button. Otherwise select the (**Separator:**) **Character** radio button and enter the character in the neighbouring text box. Set the delimiter in the **Delimiter:** field, and check the **Strict quotes** button if you want to ignore any data that hasn't been delimited.

You can also specify the escape character. This character can be used to escape the delimiter character if it occurs in any of the fields. Since the escape character is a backslash (\ ) by default, this means that if the data contains any (La)TeX commands the backslash will need to be doubled. This conflict can be avoided by changing the CSV escape character to something else (that doesn't occur in your data). To change it, select the (**Escape:**) **Character** button and enter the character in the neighbouring text box. Alternatively, you can suppress the CSV escape character, in which case the delimiter character can't occur within the data. To do this, select the **None** button. The character encoding can be changed through the **Encoding:** drop-down box.

Check the **Has Header Row** button if your CSV files have a header row otherwise uncheck it, and check the **Skip empty rows** button to skip empty rows. To skip a set number of rows, change the **Number of lines to skip at the start** value to the required number of rows. (Use 0 to switch off this function.) Note that the skip lines function is independent of the skip empty rows. If you have set the skip lines value to, say, 3 then the first 3 lines are automatically skipped regardless of whether or not they have any content. The check for empty rows won't start until the next row (row 4, in this case). The header, skip empty rows and skip lines settings are also used by the spreadsheet import functions.

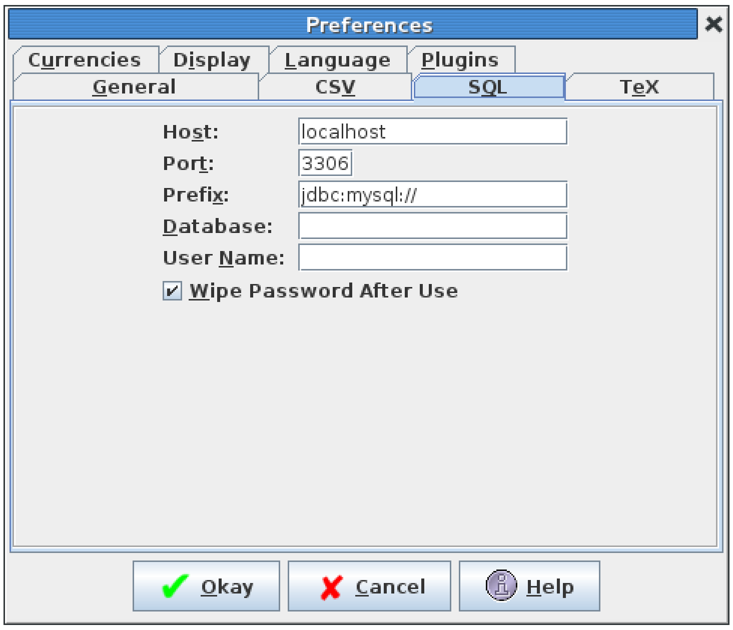
7.3 SQL Connection

In the **SQL Connection** tab (Figure 7.3), you can specify the SQL connection information. Enter the host name and port number the SQL server is running on in the **Host:** and **Port:** fields. Currently, the only available prefix is "`jdbc:mysql://`", which is the JDBC driver for MySQL. If you are using another driver or SQL database, you'll have to add the relevant library to the `lib` directory and add it to the class path used by `datatooltk.jar`. Enter the name of the database you want to connect to in the **Database:** field and the associated user name in the **User Name:** field. If you want the password wiped from memory as soon as a connection has been made, make sure the **Wipe Password After Use** box has been selected.

7.4 TeX I/O

The **TeX I/O** tab (Figure 7.4) governs TeX related settings. LaTeX is no longer used to help `datatooltk` import data from a `probsoln` dataset. Instead the TeX Parser Library is used to parse `tex` files.

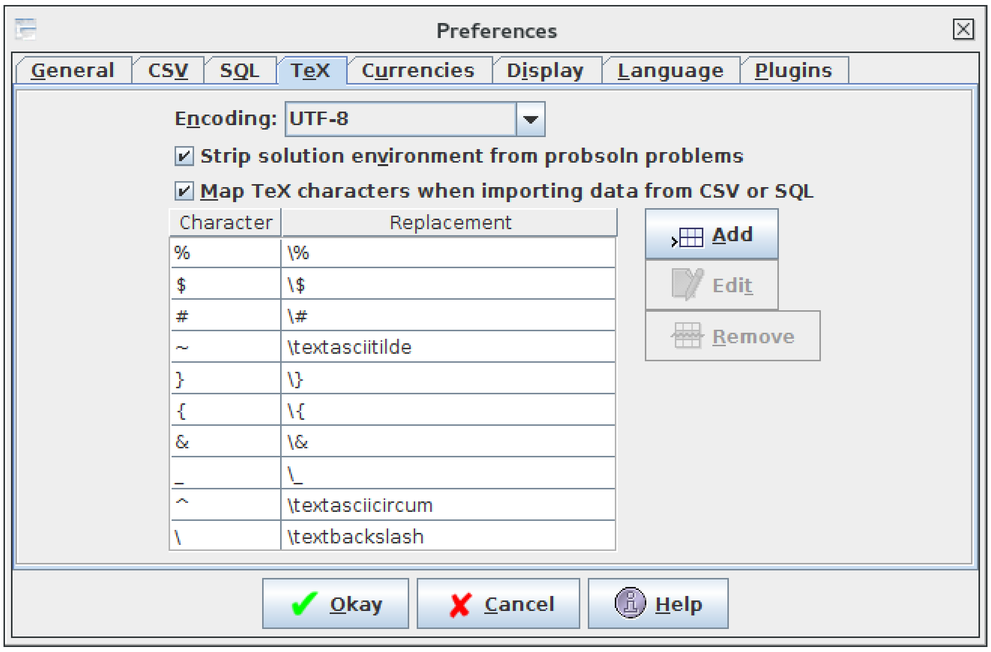
The default file encoding for TeX files (including `dbtex` and `dtltex` files opened and saved by `datatooltk`) is set in this tab in the **Default Encoding** dropdown list. Note that the `dbtex` and `dtltex` file formats allow a special comment on the first line that identifies not only the



Host:	localhost
Port:	3306
Prefix:	jdbc:mysql://
Database:	
User Name:	

☑ Wipe Password After Use

Figure 7.3: SQL Tab



Encoding: UTF-8

☑ Strip solution environment from probsoln problems

☑ Map TeX characters when importing data from CSV or SQL

Character	Replacement
%	\%
$	\$
#	\#
~	\textasciitilde
}	\}
{	\{
&	\&
_	_
^	\textasciicircum
\	\textbackslash

Figure 7.4: TeX Tab

file format but also the encoding. For example, a DBTEX 3.0 file that's UTF-8 encoded should start with the line:

```
% DBTEX 3.0 UTF-8
```

The encoding identifier may also be an inputenc option name, such as `utf8`.

As from datatool v3.0, data can be input into a document using \DTLread, which includes support for dbtex and dtltex file formats: DBTEX 3.0, DBTEX 2.0, DTLTEX 3.0 and DTLTEX 2.0. Earlier versions of datatool only support DBTEX 2.0 and DTLTEX 2.0. These formats are described in the datatool v3.0+ manual.

The DBTEX 3.0 format is the only one that supports the datum format, where the textual representation, numeric value, currency symbol and data type are stored separately, which means that the textual representation doesn't need to by parsed by datatool in numeric contexts. This is therefore the default for new databases.

The dtltex formats can have their elements converted to datum format while the file is read by \DTLread, depending datatool's store-datum setting. However, this setting doesn't affect the dbtex formats.

No format allows verbatim or category code changes to occur in the data. The entries should only contain document commands, not internal @-commands or LaTeX3 syntax. (The entries can, of course, contain commands which use internals in their definition, but any arguments must assume the default category codes.)

In general it's best to update datatool at the same time as updating datatooltk, but if you have an older version of datatool that doesn't support the newer file formats, then you can change the default format in the **Default Format** selector to DBTEX 2.0 to allow for backward compatibility. This will be used for new databases or databases obtained by importing from other sources. If you save a database that was loaded from a dbtex or dtltex file, then the original format will be the default unless the **Override DBTEX/DTLTEX input format** checkbox is selected.

If a database is saved in the DBTEX 3.0 format, then you can specify whether or not to save the datum elements. In general, it's best to use the datum format for decimals and currency. It can also be useful for integers if their numeric values need to be used in a calculation in the document. However, if the integer values are unique identifies that need to be queried, it's better not to use the datum format as it can interfere with queries. There's usually no need to save strings in the datum format unless a column or row contains mixed types that need parsing. (For example, with column or row aggregate actions.) As with a column containing unique integer value identifiers, a column containing unique string identifiers that need to be queried are best saved without the datum format.

To indicate whether or not to used the datum format when saving to a DBTEX 3.0 file for entries, select the applicable radio button: **no entries** (don't use the datum format for any entries), **all non-null values** (use the datum format for all non-null entries), **according to**

header type (determine whether or not to use the datum format according to the column's data type), or **according to element type** (determine whether or not to use the datum format according to the entry's data type, which may be different to the column type).

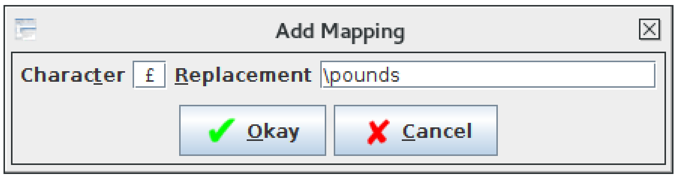
For the last two options, the applicable type (column or entry) determines whether or not to use the datum format. If either of those options are set, the following checkboxes will become visible:

- **Integer** use the datum format for integers, if this box is selected;
- **Decimal** use the datum format for decimals, if this box is selected;
- **Currency** use the datum format for currencies, if this box is selected;
- **String** use the datum format for currencies, if this box is selected.

Null entries are omitted when saving the data.

When importing `probsoln` data, you may find it more convenient to strip any instances of the solution environment, particularly the implicit use of this environment by the unstarred version of `\newproblem`. You can now choose whether or not to omit `\begin{solution}` and `\end{solution}` by selecting the **Strip solution environment from probsoln problems** button. If checked, any instances of solution contained within definitions (provided by `\newproblem` or defproblem) will be removed.

In the **TeX I/O** tab you can also specify whether non-TeX sources contain literal content or TeX code. If the **Literal** box is selected, then the mapping table and buttons will become visible. This table will be used to map the identified characters to the given commands. To add another mapping, click on the **Add** button, which opens the dialog box shown in Figure 7.5. Note that this table is only designed to map single characters. You will need to use the search and replace function to perform more complex changes.

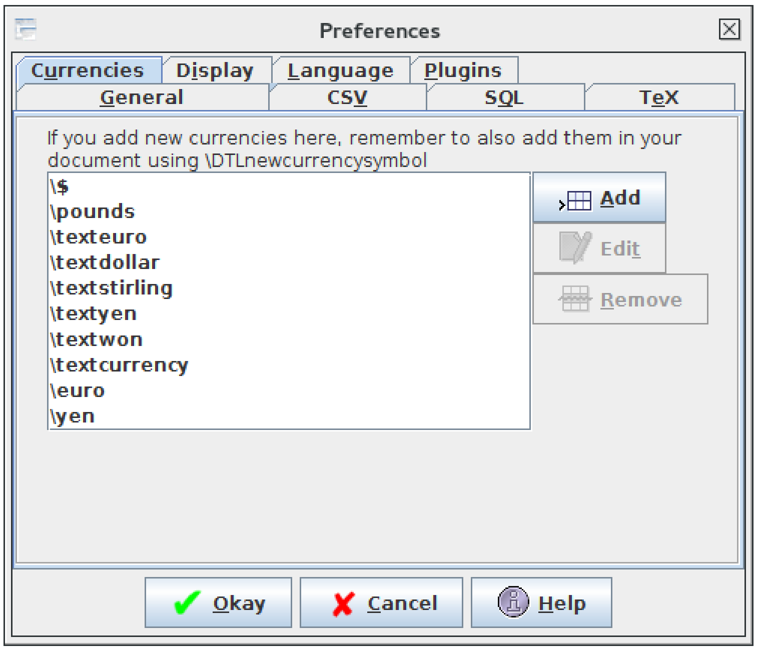


Character £ Replacement \pounds

Figure 7.5: Add Mapping Dialog

To remove a mapping, select the unwanted mapping and click on **Remove**. To edit a mapping, select the mapping and click on **Edit**.

7.5 Currencies



If you add new currencies here, remember to also add them in your document using \DTLnewcurrencysymbol

```
\$
\pounds
\texteuro
\textdollar
\textstirling
\textyen
\textwon
\textcurrency
\euro
\yen
```

Figure 7.6: Currencies Tab

If you want to identify a column as a currency type, you must make sure that datatooltk recognises the LaTeX command to typeset your currency. Known currency commands are listed in the **Currencies** tab (Figure 7.6). If you add any currencies to the list, remember to add them in your document as well with `\DTLnewcurrencysymbol`.

7.6 Cell Editor Settings

The **Cell Editor Settings** tab governs the font and style for the cell editor window (see §3.1).

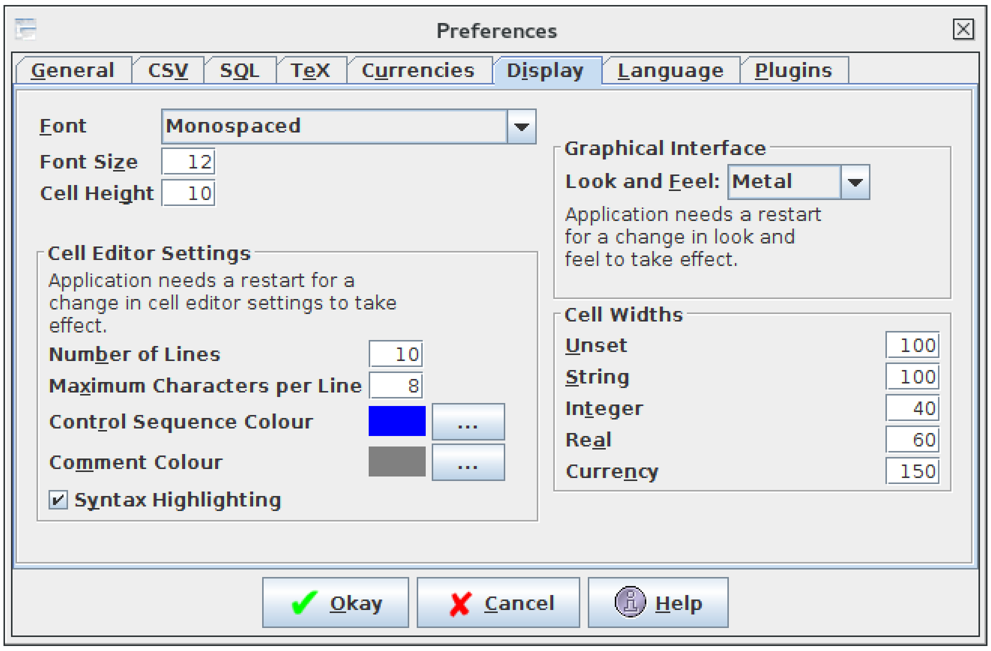
The default font used in cell entries is a monospaced font. This can be changed using the **Font** drop-down menu. You can also set the font size in the **Font Size** field. The text colornd background coloran be changed using the **Text Color** and **Background** selectors.

Simple syntax highlighting can be enabled or disable by selecting or deselecting the **Syntax Highlighting** checkbox. Enabling this option will show the **Comment Color** and **Control Sequence Color** selectors for comment lines and control sequences.

The sample area at the top of the panel shows the style according to the current selection.

The preferred dimensions of the editor pane on startup can also be set with the **Preferred Height** and **Preferred Width** widgets. Note that this is a preference but it may not be possible to set the given dimensions. For example, the window may be wider in order to accommodate the toolbar.

7.7 Display



Preferences

General | CSV | SQL | TeX | Currencies | Display | Language | Plugins

Font	Monospaced
Font Size	12
Cell Height	10

Cell Editor Settings

Application needs a restart for a change in cell editor settings to take effect.

Number of Lines	10
Maximum Characters per Line	8
Control Sequence Colour	🟦 …
Comment Colour	⬛ …

☑ Syntax Highlighting

Graphical Interface

Look and Feel: Metal

Application needs a restart for a change in look and feel to take effect.

Cell Widths

Unset	100
String	100
Integer	40
Real	60
Currency	150

Okay | Cancel | Help

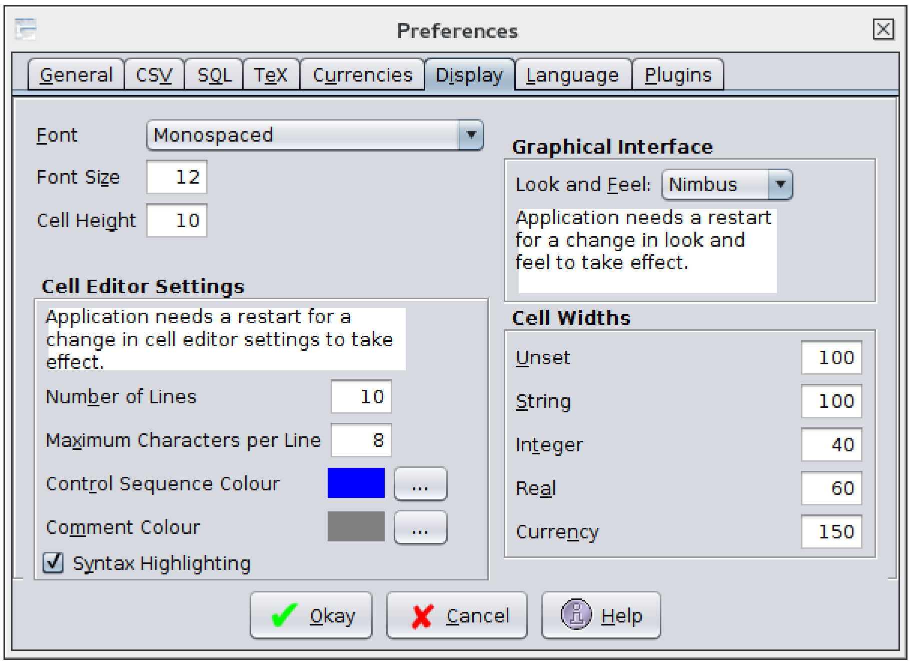
Figure 7.7: Display Tab

The **Display** (Figure 7.7) tab governs user interface settings. (Use the **Cell Editor Settings** tab for the cell editor display settings.) By default, each string cell has a maximum of four lines visible in the main window. (Real and integer columns only have a single line visible.) This number can be changed in the **Cell Height** field. Each column has a default width that depends on the data type for that column. The values are listed in the **Table Cell Dimensions** area. These can be changed as required.

The Look and Feel (L&F) refers to the way the graphical interface is rendered. You can use the drop-down menu to select a different L&F, but you need to restart `datatooltk` for the change to take effect. For example, Figure 7.7 shows the "Metal" L&F whereas Figure 7.8 shows the same window but with the "Nimbus" L&F.

7.8 Localisation

The **Localisation** tab (Figure 7.9) controls the localisation settings. The language used by the manual accessed via **Help** ➜ **Manual** can be set from the **Manual Language:** drop-down list. The language used in the messages, menu items, buttons and GUI labels can be set from the **GUI Language** drop-down list. Note that you have to restart `datatooltk` for these changes to take effect.



Preferences

Font: Monospaced
Font Size: 12
Cell Height: 10

Cell Editor Settings

Application needs a restart for a change in cell editor settings to take effect.

Number of Lines: 10
Maximum Characters per Line: 8
Control Sequence Colour: 🟦
Comment Colour: ⬛
☑ Syntax Highlighting

Graphical Interface

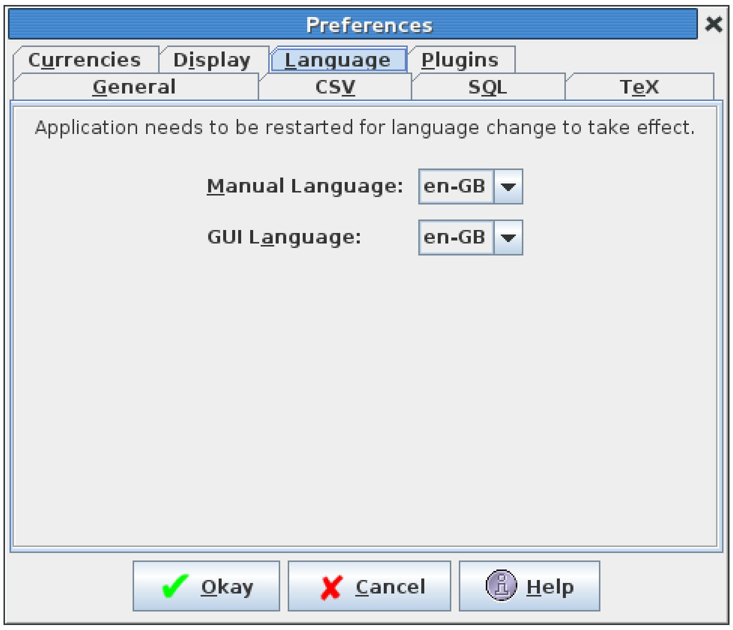
Look and Feel: Nimbus

Application needs a restart for a change in look and feel to take effect.

Cell Widths

Unset	100
String	100
Integer	40
Real	60
Currency	150

Figure 7.8: Display Tab (Nimbus Look and Feel)



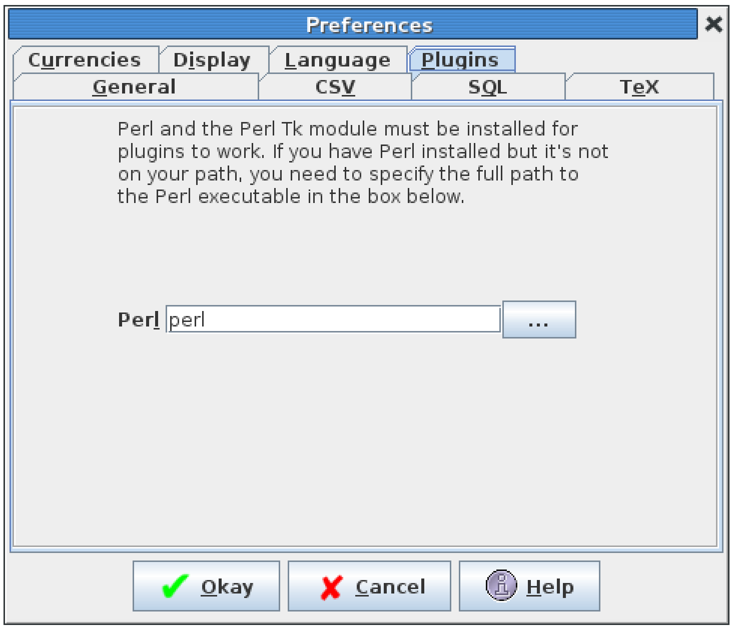
Preferences

Application needs to be restarted for language change to take effect.

Manual Language: en-GB
GUI Language: en-GB

Figure 7.9: Language Tab

7.9 Plugins



Perl and the Perl Tk module must be installed for plugins to work. If you have Perl installed but it's not on your path, you need to specify the full path to the Perl executable in the box below.

Perl: perl ...

Figure 7.10: Plugins Tab

In order to use datatooltk plugins, you must have Perl installed (and the Perl Tk module). If the Perl executable is on your path, you can just specify it as "`perl`" in the **Perl** field of the **Plugins** tab (Figure 7.10). If it's not on your path, you will have to specify the full path name in this tab. You can use the ellipsis button to browse your filing system.

8 Help Menu (GUI Mode)

Help ➜ About

The **Help ➜ About** menu item shows the **About DatatoolTk** dialog with version details. In batch mode this information can be obtained with the `--version` switch.

Help ➜ Manual F1

The application's manual is available as either a PDF document, which can be viewed outside of the application, or as a set of HTML files which can be viewed within the application via the **Help ➜ Manual** menu item. This will open the primary help window (§8.1), but some dialog boxes may also have a **Help** button that will open a secondary help dialog (§8.2).

Both the primary help window and the secondary help dialog windows have a panel that shows a page of the manual (a help page). Note that "page" in this context refers to the HTML file displayed in the help window, which typically contains a section, and doesn't relate to the page numbers in the PDF. The HTML index page is obtained from the same source code as the PDF index page, but the locations are converted from a PDF page number to the HTML page title (preceded by the marker ☛).

Although the help page is not editable, for some versions of Java, the caret is visible when the page has the focus, and the caret can be moved around using the arrow keys on your keyboard.

Help Popup Menu

The Help Popup Menu (see Figure 8.1) can be activated on the current help page for both the primary and secondary help windows. The mouse press to show a popup menu is typically the right mouse button, but this may not be the case for all operating systems. The popup menu can also be activated using the context menu ☰ key if the help page has the focus. The menu has the following items.

Help Popup Menu ➜ **View Image**

If the popup menu is activated over an image, the **View Image** item will open the **Image Viewer** window (see §8.5) which can be used to enlarge the image. This item will be disabled if the popup menu wasn't activated over an image.

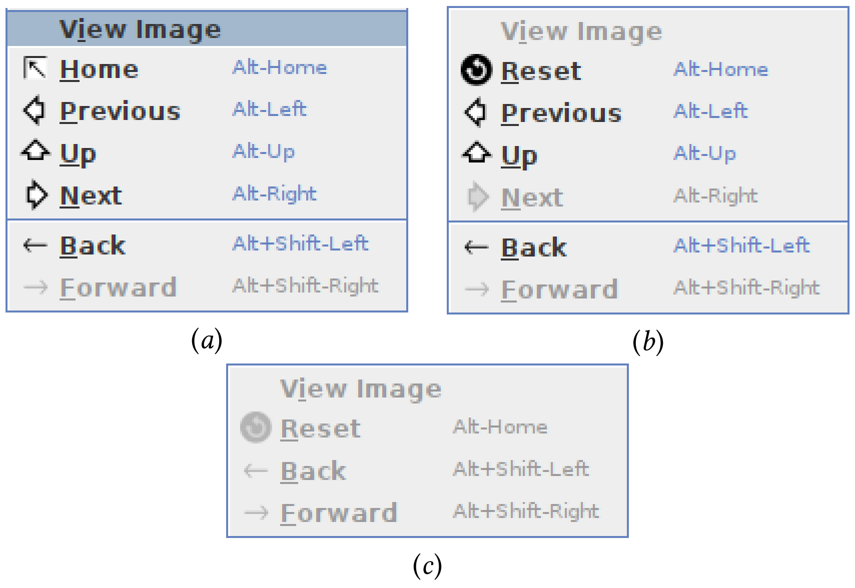


Figure 8.1: Help Page Popup Menus: (*a*) Primary Help Window; (*b*) Secondary Help Dialog with Multiple Topic Pages; (*c*) Secondary Help Dialog with Single Topic Pages

Where the popup menu was activated using the context menu ≣ key, the position of the caret will determine whether or not to enable this menu item.

≣

Help Popup Menu ➜ **Home** **Alt**+**Home**

If the popup menu is activated on the primary help window (Figure 1(*a*)), this will behave as the **Navigation** ➜ **Home** menu item (which switches the current page to the first page of the document). This menu item is not available on secondary help windows.

≣

Help Popup Menu ➜ **Reset** **Alt**+**Home**

If the popup menu is activated on the secondary help dialog (figures 1(*b*) & 1(*c*)), this will behave as the **Navigation** ➜ **Reset** menu item (which switches the current page back to the relevant page or the first in the applicable section of the dialog topic). It will be disabled if the current page is the reset target page. This menu item is not available on the primary help window.

≣

Help Popup Menu ➜ **Up** **Alt**+**↑**

If the popup menu is activated on the primary help window or on the secondary help window that has multiple pages (figures 1(*a*) & 1(*b*)), then this will behave as the primary **Navigation**

➜ **Up** or secondary **Navigation** ➜ **Up** menu items. (That is, it will move up a hierarchical level, if available.) This menu item will be disabled if there is no parent page (or, for secondary windows, no parent page within the topic set).

≡

Help Popup Menu ➜ **Previous** Alt + ←

If the popup menu is activated on the primary help window or on the secondary help window that has multiple pages (figures 1(*a*) & 1(*b*)), then this will behave as the primary **Navigation** ➜ **Previous** or secondary **Navigation** ➜ **Previous** menu items. (That is, it will move to the previous page, if available.) This menu item will be disabled if there is no previous page (or, for secondary windows, no previous page within the topic set).

≡

Help Popup Menu ➜ **Next** Alt + →

If the popup menu is activated on the primary help window or on the secondary help window that has multiple pages (figures 1(*a*) & 1(*b*)), then this will behave as the primary **Navigation** ➜ **Next** or secondary **Navigation** ➜ **Next** menu items. (That is, it will move to the next page, if available.) This menu item will be disabled if there is no next page (or, for secondary windows, no next page within the topic set).

≡

Help Popup Menu ➜ **Back** Alt + ⇧ + ←

This will behave as the primary **Navigation** ➜ **Back** or secondary **Navigation** ➜ **Back** menu items. (That is, it will move back a page in the history list, if available.) This menu item is in all the help page popup menus but will be disabled if there is no page to go back to.

For the secondary help windows, it's possible to follow a link in the current page to a page outside the topic set. The menu item can take you back to the previously visited page viewed in that secondary dialog window.

≡

Help Popup Menu ➜ **Forward** Alt + ⇧ + →

This will behave as the primary **Navigation** ➜ **Forward** or secondary **Navigation** ➜ **Forward** menu items. (That is, it will move forward a page in the history list, if available.) This menu item is in all the help page popup menus but will be disabled if there is no page to go forward to.

8.1 The Primary Help Window

The primary help window is the main help frame accessed via **Help** ➜ **Manual**, which has a panel that shows a page of the manual (a help page). Links in the page and the GUI navigation

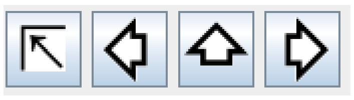


Figure 8.2: Primary Help Window Navigation Buttons (Home, Previous, Up, Next)

elements provide a way to switch to a different page.

There is a menu bar with items for navigation actions or adjusting GUI settings. Some menu items are replicated as buttons in the toolbar, which is split into different regions: navigation, lookup, settings, and history. The forward, up and next navigation actions can also be implemented by buttons in the lower navigation panel at the bottom of the window.

Navigation

The **Navigation** menu provides a way to move around the document. Figure 8.2 shows the corresponding four navigation buttons in the toolbar: Home Page (go to the start of the manual), Previous Page (go to the previous section), Up (go to parent section), and Next Page (go to the next section).

Navigation ➜ Home Alt+Home

The **Navigation ➜ Home** item, which is also available as a button on the toolbar, will replace the current view with the first page of the document.

Navigation ➜ Up Alt+↑

The **Navigation ➜ Up** item, which is also available as a button on the toolbar, will replace the current view with the parent page of the current hierarchical level. The item and button will be disabled if there is no parent page (that is, if the current page is the document's home page). The parent page may also be the previous page if the current page is the first in its current hierarchical level.

Navigation ➜ Previous Alt+←

The **Navigation ➜ Previous** item, which is also available as a button on the toolbar, will replace the current view with the previous page. The item and button will be disabled if there is no previous page. (That is, if the current page is the first page of the document.)

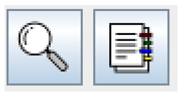


Figure 8.3: Search and Index Buttons

Navigation ➜ Next Alt+➔

The **Navigation ➜ Next** item, which is also available as a button on the toolbar, will replace the current view with the next page. The item and button will be disabled if there is no next page. (That is, if the current page is the last page of the document.)

Figure 8.3 shows the search and index buttons, which may be used to lookup relevant pages.

Navigation ➜ Search…

The **Navigation ➜ Search…** item, which is also available as a button on the toolbar, will open the **Search** window (see §8.4), from which you can search the document for a keyword.

Navigation ➜ Index…

The **Navigation ➜ Index…** item, which is also available as a button on the toolbar, will open the index page in a separate window (see Figure 8.4). You can also open the same page in the help window at the end of the document. The separate index window provides a way of navigating the document without having to keep returning to the index page. Additionally, the index window has a split page with links on the left to scroll the page to a letter group.

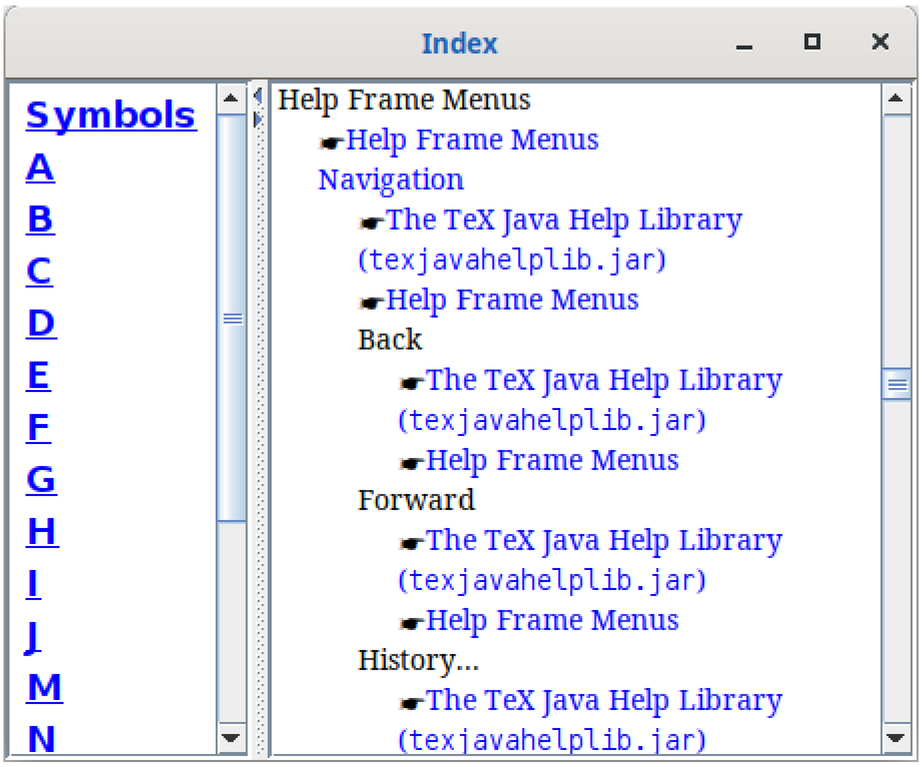
If an indexed item is shown as a hyperlink, then that link will go to the principle definition of that item. The indexed item may also be followed by a list of pertinent locations that are preceded by the symbol ☛.

Figure 8.5 shows the history buttons. Note that the forward button is greyed (disabled) because the currently viewed page is at the end of the history list, so it's not possible to go forward.

Navigation ➜ History… Ctrl+⇧+H

The **Navigation ➜ History…** menu item, which is also available as a button on the toolbar, opens the **Page History** window, (see Figure 8.6).

The current page has the title shown in bold and is preceded by the symbol ☞. Select the required page and click on the **Go** button.



Help Frame Menus
☛Help Frame Menus
Navigation
☛The TeX Java Help Library (texjavahelplib.jar)
☛Help Frame Menus
Back
☛The TeX Java Help Library (texjavahelplib.jar)
☛Help Frame Menus
Forward
☛The TeX Java Help Library (texjavahelplib.jar)
☛Help Frame Menus
History...
☛The TeX Java Help Library (texjavahelplib.jar)

Figure 8.4: Index Window

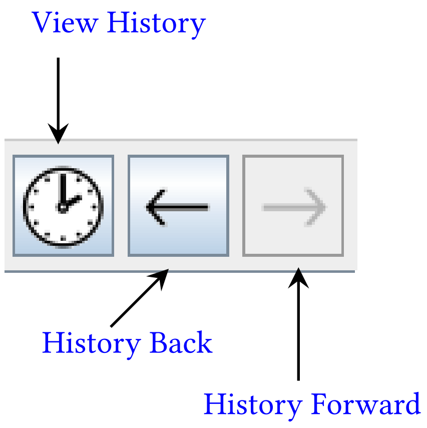
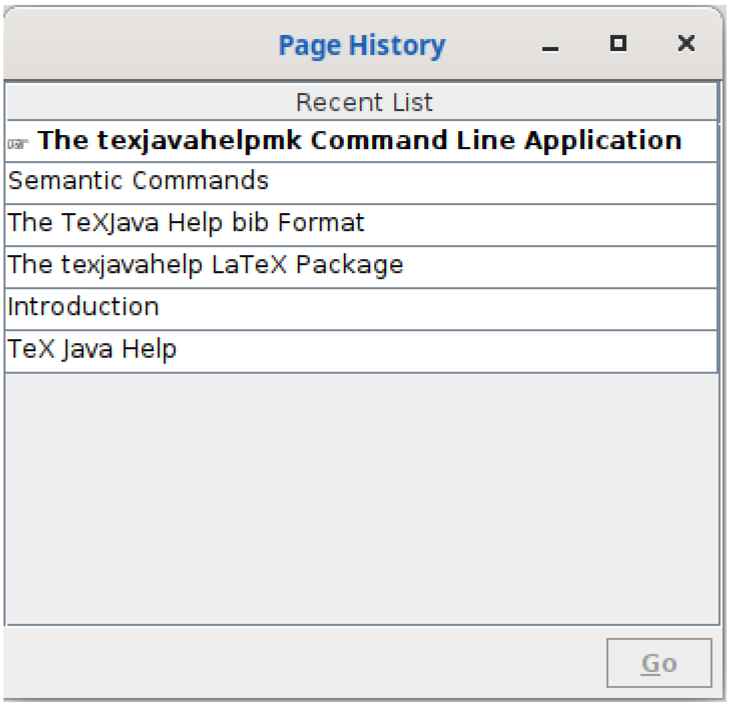


Figure 8.5: History Buttons


Page History

Recent List

The texjavahelpmk Command Line Application

Semantic Commands

The TeXJava Help bib Format

The texjavahelp LaTeX Package

Introduction

TeX Java Help

Figure 8.6: The Page History Window

Navigation ➜ Back Alt+⇧+←

The **Navigation ➜ Back** menu item, which is also available as a button on the toolbar, will replace the current view with the previously viewed page from this history list. The item and button will be disabled if there is no previously viewed page.

Navigation ➜ Forward Alt+⇧+→

The **Navigation ➜ Forward** menu item, which is also available as a button on the toolbar, will replace the current view with the next page in the history list. The item and button will be disabled if the currently viewed page is at the end of the history list.

Settings

The **Settings** menu can be used to change the graphical interface settings. These settings affect the primary and secondary help windows, as well as some other related windows. Note that this is separate from the main application settings.

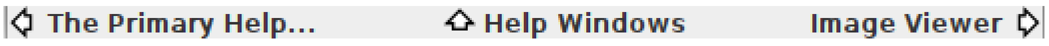


Figure 8.7: Help Page Lower Navigation Bar

Settings ➜ **Font Decrease** NumPad -

The **Settings** ➜ **Font Decrease** item decreases the font size by 1.

Settings ➜ **Font Increase** NumPad +

The **Settings** ➜ **Font Increase** item increases the font size by 1.

Settings ➜ **Font...** Ctrl + F

The **Settings** ➜ **Font...** item opens the **Help Page Font** dialog (see §8.3).

Settings ➜ **Lower Navigation...**

The **Settings** ➜ **Lower Navigation...** item opens the **Lower Navigation Settings** dialog. This governs the lower navigation bar (see Figure 8.7) along the bottom of the primary help window, which has smaller previous, up and next buttons. These buttons by default have the corresponding page titles next to them, but they will be truncated if they exceed the limit. This limit can be changed with the **Maximum number of characters** widget. Alternatively, you can hide the text by deselecting the **Include text** checkbox.

8.2 Secondary Help Window

The secondary help windows are more minimalist and will only show the relevant help page or set of pages that are applicable to the context that was used to open the secondary help window. If only one page is applicable, there won't be a navigation tree, otherwise the navigation tree will only show the applicable pages.

The search, history and index windows are unavailable, but it is possible to move back and forward in the history list for the current secondary help window. The topic page will be added to the primary help window history but otherwise the page history lists aren't shared between the help windows.

The secondary help windows are designed for use with modal dialogs (that is, a window that blocks the main application window) to provide help for the particular dialog. The primary help window can't be accessed while a modal dialog is open so it will automatically be closed when a secondary help window is opened. You can re-open the primary help window once you have closed the modal dialog.

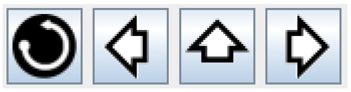


Figure 8.8: Secondary Help Window Navigation Buttons for Multi-Page Topics (Reset, Previous, Up, Next)

Navigation

The **Navigation** menu provides a way to move around the topic pages.

Navigation ➜ Reset Alt+Home

The **Navigation ➜ Reset** item switches the current page to the first page of the context topic. This menu item will be disabled if the current page is the reset target page.

Navigation ➜ Back Alt+⇧+←

The **Navigation ➜ Back** goes back to the previously visited page. Note that the history is specific to the current secondary help dialog instance and does not include the history from the primary help window. This menu item will be disabled if there is no page in the history list to go back to.

Navigation ➜ Forward Alt+⇧+→

The **Navigation ➜ Forward** moves forward in the history list, if applicable. This menu item will be disabled if there is no page in the history list to go forward to.

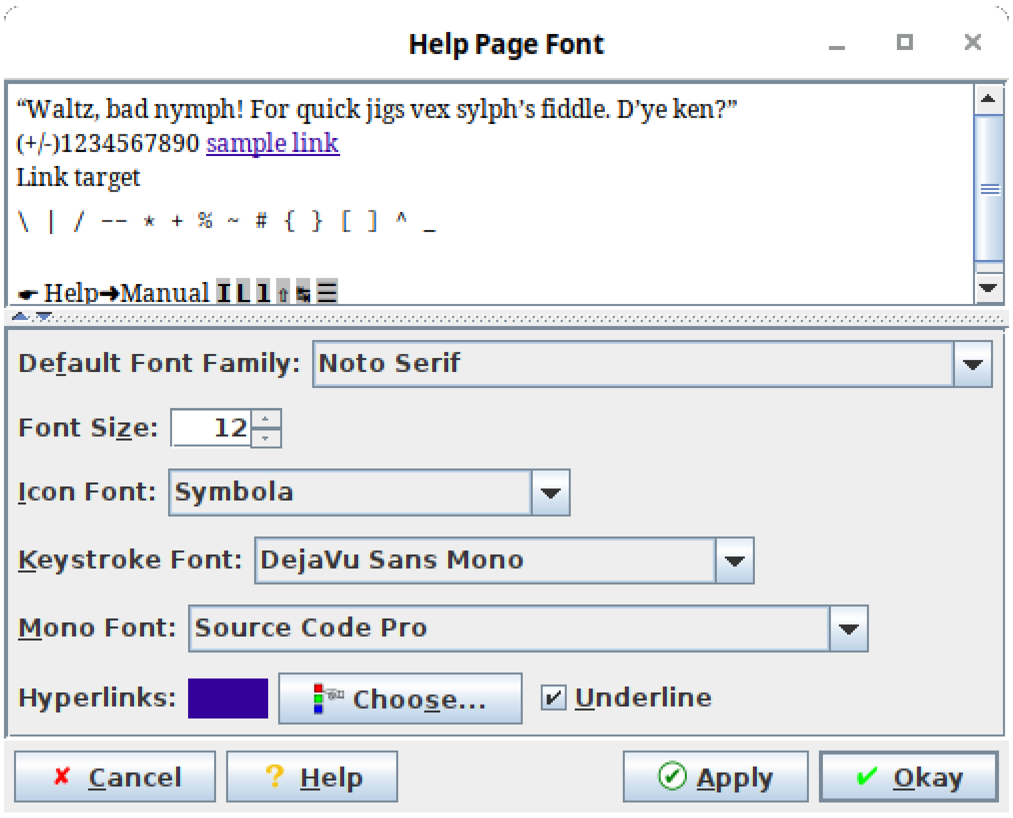
The Previous Page, Up and Next Page buttons (Figure 8.8) are only available if the topic context contains multiple pages.

Navigation ➜ Previous Alt+←

The **Navigation ➜ Previous** menu item is only available if there are multiple pages for the topic context and will switch the current page with the previous page in the topic set. This menu item will be disabled if the previous page is not within the topic set.

Navigation ➜ Up Alt+↑

The **Navigation ➜ Up** menu item is only available if there are multiple pages for the topic


Help Page Font

"Waltz, bad nymph! For quick jigs vex sylph's fiddle. D'ye ken?"
(+/-)1234567890 sample link
Link target
\ | / -- * + % ~ # { } [ ] ^ _

☛ Help➜Manual

Default Font Family: Noto Serif

Font Size: 12

Icon Font: Symbola

Keystroke Font: DejaVu Sans Mono

Mono Font: Source Code Pro

Hyperlinks: Choose... ✓ Underline

Figure 8.9: Help Page Font Dialog

context and will switch the current page with the parent page if it's within the topic set. This menu item will be disabled if there is no parent page or if the parent page is not in the topic set.

Navigation ➜ Next ☰ Alt+➔

The **Navigation ➜ Next** menu item is only available if there are multiple pages for the topic context and will switch the current page with the next page in the topic set. This menu item will be disabled if the next page is not within the topic set.

8.3 Help Font Dialog

The **Settings ➜ Font...** item opens the **Help Page Font** dialog (see Figure 8.9). Use the **Default Font Family** selector for the main body font family and the **Font Size** selector for the main body font size. Icon characters, such as ☞, may not be available for your preferred font family, so you can specify an alternative with the **Icon Font** selector. This will only list fonts that support some commonly used icon characters.

Use the **Keystroke Font** selector to choose the font to show keystrokes (such as ⇧) and the **Mono Font** selector to choose the font to display code fragments (such as `% \ { } #`).

The document hyperlink style can also be changed with the **Hyperlinks Choose...** and **Underline** widgets.

The styles are applied to the primary help window, all secondary help windows and related windows, such as the **Page History** or index windows.

8.4 Searching the Documentation

The **Search** window (which can be opened from the primary help window with **Navigation ➜ Search...**) provides a way to search the documentation. Enter the desired search term or terms into the **Keywords** box. Select the **Case-sensitive** checkbox for a case-sensitive search and the **Match whole word** checkbox for an exact match. If the **Match whole word** checkbox is not selected, the search will be slower and will match any instances of the keyword appearing as a sub-string of other words as well as whole-word matches.

The search is performed by looking up a pre-compiled set of words with associated locations that was created when the documentation was built. It's not possible to search for exact phrases. The results are ordered according to the number of matches found in each block or paragraph.

Click on the **Find** button to the right of the **Keywords** box or use the **Search ➜ Find** menu item to start searching. Note that small common words, such as "and", will be ignored.

If any matches are found, the title of the relevant page is shown as a hyperlink, which links to the start of the page. The title is followed by a block of text where the search term (or terms) was found (which will be highlighted, as shown in Figure 8.10). Clicking on the block of text should scroll to a nearby location in the relevant page.

Search

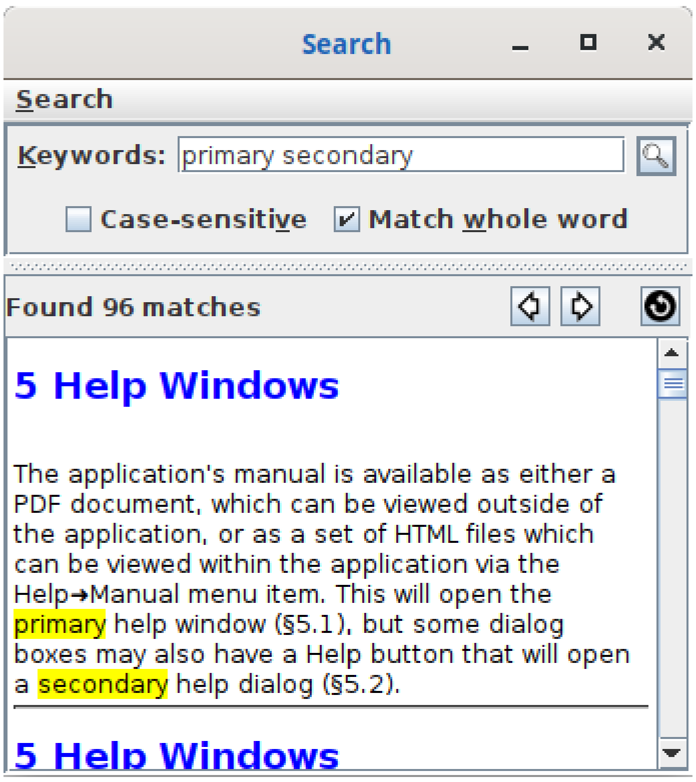
The **Search** menu has the following menu items.

Search ➜ Find

The **Search ➜ Find** menu item starts searching for the given keywords. An error box will be displayed if no keywords have been supplied.

Search ➜ Previous Result Control+⇧+F

The **Search ➜ Previous Result** menu item will scroll the result list to the previous result.



5 Help Windows

The application's manual is available as either a PDF document, which can be viewed outside of the application, or as a set of HTML files which can be viewed within the application via the Help→Manual menu item. This will open the primary help window (§5.1), but some dialog boxes may also have a Help button that will open a secondary help dialog (§5.2).

5 Help Windows

Figure 8.10: Search Window

Search ➜ Next Result Control + F

The **Search ➜ Next Result** menu item will scroll to the result list to the next result.

Search ➜ Clear ⇧ + Esc

The **Search ➜ Clear** menu item will clear the current result list and the keyword search box so that you can perform a new search.

Search ➜ Stop Esc

The **Search ➜ Stop** menu item can be used to stop a search if it's taking too long to complete.

8.5 Image Viewer

The **View Image** item in the Help Popup Menu for both the primary and secondary help windows will be enabled if the Help Popup Menu is activated over an image. The **View Image** item will open the image in the **Image Viewer** window. If the image had alt text specified, this will be displayed in the area above the image.

Within the **Image Viewer** window, the image can be enlarged using the **Magnify (%)** spinner. The up and down spinner controls go in steps of 25 (as opposed to the **Increase** and **Decrease** action, which have an increment of 5). Alternatively, press the shift key ⇧ and drag the mouse to select an area to zoom in on. Be sure to keep the shift key down when you release the mouse. If you change your mind, release shift before releasing the mouse button. If the shift key isn't pressed when you initiate the drag, dragging will scroll the image instead. Double-clicking the mouse on the image will go back to the previous magnification.

Image Viewer Menu

The Image Viewer Menu is a popup menu that can be activated anywhere over the image in the **Image Viewer** window. The following menu items are available.

Image Viewer Menu ➜ **Fit to Width**

The **Fit to Width** item will scale the image so that it fits the window width. This action has a corresponding button on the toolbar.

Image Viewer Menu ➜ **Fit to Height**

The **Fit to Height** item will scale the image so that it fits the window height. This action has a corresponding button on the toolbar.

Image Viewer Menu ➜ **Fit to Page**

The **Fit to Page** item will scale the image so that it fits within the window area. This action has a corresponding button on the toolbar.

Image Viewer Menu ➜ **Increase** Alt + ↑

The **Increase** item will increase the current magnification. This action has a corresponding button on the toolbar.

Image Viewer Menu ➜ **Decrease** Alt + ↓

The **Decrease** item will decrease the current magnification. This action has a corresponding button on the toolbar.

Image Viewer Menu ➜ **100%**

The **100%** item will set the magnification factor to 100%. This action has a corresponding button on the toolbar.

Image Viewer Menu ➜ **200%**

The **200%** item will set the magnification factor to 200%.

Image Viewer Menu ➜ **500%**

The **500%** item will set the magnification factor to 500%.

9 License

DatatoolTk is licensed under the terms of the GNU General Public License version 3 (GPLv3). `datatooltk` depends on the following third party libraries whose jar files are in the `lib` directory:

- TeX Parser Library `texparserlib.jar` (GPL, `https://github.com/nlct/texparser`);
- TeX Java Help `texjavahelplib.jar` (GPL, `https://github.com/nlct/texjavahelp`);
- MySQL connector `mysql-connector-java.jar` (GPLv2, `http://dev.mysql.com/downloads/connector/j/`);
- Apache POI `poi-5.2.5.jar`, `poi-ooxml-5.2.5.jar` and `poi-ooxml-full-5.2.5.jar` (Apache 2, `https://poi.apache.org/`).

Summary of `datatooltk` Switches

`--auto-trim-labels`
Automatically strip leading and trailing spaces from database and column identifiers.

`--batch` (or `-b`)
Invoke `datatooltk` in batch mode.

`--compat` ⟨*level*⟩
Sets the compatibility level, where ⟨*level*⟩ may be "`latest`" or "`1.6`".

`--csv` ⟨*csv file*⟩
Import data from the given CSV file.

`--csv-delim` ⟨*character*⟩
Specified the character used to delimit values in CSV files.

`--csv-encoding` ⟨*encoding*⟩
Sets the encoding for the CSV files.

`--csvencoding` ⟨*encoding*⟩
Synonym for `--csv-encoding`.

`--csv-escape` ⟨*character*⟩
Specified the character used to escape values in CSV files.

`--csvescape` ⟨*character*⟩
Synonym for `--csv-escape`.

`--csv-header`
The CSV and spreadsheet files have a header row (default).

`--csvheader`
Synonym for `--csv-header`.

`--csv-sep` ⟨*character*⟩
Specified the character used to separate values in CSV files.

`--csv-skip-empty-rows`
Skip empty lines found in CSV and spreadsheet files.

`--csv-skiplines` ⟨*n*⟩

Skip the first ⟨*n*⟩ rows in CSV files. Blank rows are always included in this count, even if `--csv-skip-empty-rows` is set.

`--csv-strictquotes`

Ignore any undelimited information in CSV files.

`--debug`

Enables debug mode.

`--delim` ⟨*character*⟩

Synonym for `--csv-delim`.

`--filter` ⟨*key*⟩ ⟨*operator*⟩ ⟨*value*⟩

Add the given filter, which returns true if the value in the column whose label is given by ⟨*key*⟩ matches the operation ⟨*operator*⟩ ⟨*value*⟩ where ⟨*operator*⟩ may be one of: "`eq`" (equals ⟨*value*⟩), "`ne`" (does not equal ⟨*value*⟩), "`le`" (less than or equal ⟨*value*⟩), "`lt`" (less than ⟨*value*⟩), "`ge`" (greater than or equal ⟨*value*⟩), "`gt`" (greater than ⟨*value*⟩), "`regex`" (matches the regular expression ⟨*value*⟩). Multiple filters may be specified.

`--filter-and`

Use logical AND operator when filtering.

`--filter-exclude`

When filtering, discard rows that match the filter and keep those that don't match (default).

`--filter-include`

When filtering, discard rows that don't match the filter and keep those that do match (default).

`--filter-or`

Use logical OR operator when filtering (default).

`--gui` (or -g)

Invoke `datatooltk` in GUI mode.

`--help` (or -h)

Prints help and exits.

`--in` ⟨*filename*⟩ (or -i)

Load the `datatool` file ⟨*filename*⟩.

`--literal`

`--map-tex-specials`

Map TEX special characters when importing data from non-TeX sources (that is, from CSV, spreadsheets or databases).

`--merge` ⟨*key*⟩ ⟨*db file*⟩

Merge the input or imported database with the database stored in the given ⟨*db file*⟩.

`--merge-csv` ⟨*key*⟩ ⟨*csv file*⟩

As `--merge` but the data to be merged is imported from the given CSV file.

`--merge-ods` ⟨*key*⟩ ⟨*ods file*⟩

As `--merge` but the data to be merged is imported from the given Open Document spreadsheet `ods` file.

`--merge-probsoln` ⟨*key*⟩ ⟨*filename*⟩

As `--merge` but the data to be merged is imported from the given `probsoln` file.

`--merge-sql` ⟨*key*⟩ ⟨*query statement*⟩

As `--merge` but the data to be merged is imported using the given SQL query.

`--merge-xls` ⟨*key*⟩ ⟨*xls file*⟩

As `--merge` but the data to be merged is imported from the given Excel `xls` file.

`--name` ⟨*label*⟩

Sets the database name (that is, its identifying label) to ⟨*label*⟩.

`--noauto-trim-labels`

Don't automatically strip leading and trailing spaces from database and column identifiers.

`--noconsole-action` ⟨*action*⟩

The action to perform if `datatooltk` is run in batch mode and a password is required, where ⟨*action*⟩ may be one of: "`error`", "`stdin`", or "`gui`".

`--nocsv-escape` ⟨*character*⟩

Don't have a designated character used to escape values in CSV files.

`--nocsvescape` ⟨*character*⟩

Synonym for `--nocsv-escape`.

`--nocsv-header`

The CSV and spreadsheet files do not have a header row (default).

`--nocsvheader`

Synonym for `--nocsv-header`.

`--nocsv-skip-empty-rows`

Don't skip empty lines found in CSV and spreadsheet files.

`--nocsv-strictquotes`

Allow undelimited data in CSV files.

--nodebug

Disables debug mode (default).

--noliteral

--nomap-tex-specials

Don't map TeX special characters when importing data (default).

--noowner-only

Don't change the read/write permissions when saving `dbtex` files.

--noshuffle

Don't shuffle the database (default).

--nowipepassword

Don't wipe the password from memory as soon as it has been used to connect to an SQL database (default).

--ods ⟨*ods file*⟩

Import data from the given Open Document spreadsheet `ods` file.

--output ⟨*filename*⟩ (or `-o`)

Save the database to ⟨*filename*⟩ (batch mode only).

--output-format ⟨*format*⟩

Sets the default file format for output files.

--owner-only

Set the read/write permissions when saving `dbtex` files to owner only (operating system dependent).

--probsoln ⟨*filename*⟩

Import `probsoln` data from ⟨*filename*⟩.

--remove-columns ⟨*column list*⟩

Remove the columns identified by ⟨*column list*⟩.

--remove-except-columns ⟨*column list*⟩

--seed ⟨*number*⟩

Set the random generator seed to ⟨*number*⟩ or clear it if ⟨*number*⟩ is `""` (that is, an empty string).

--sep ⟨*character*⟩

Synonym for `--csv-sep`.

`--sheet` ⟨*sheet id*⟩

The sheet to select when reading data from an Excel workbook or Open Document Spreadsheet, where ⟨*sheet id*⟩ is either the sheet index (starting from `0`) or the name of the sheet.

`--shuffle`

Shuffle the database (always performed after `--sort`, regardless of the option order, if both are present).

`--sort [`⟨*prefix*⟩**`]`**⟨*field*⟩

Sort the database according to the column whose label is ⟨*field*⟩. Optionally, ⟨*prefix*⟩ may be `+` (ascending order) or `-` (descending order). If ⟨*prefix*⟩ is omitted, ascending is assumed.

`--sort-case-insensitive`

Use case-insensitive comparisons when letter-sorting strings (`--sort-locale none`) or for the string comparison functions used by `--filter`.

`--sort-case-sensitive`

Use case-sensitive comparisons when letter-sorting strings (`--sort-locale none`) or for the string comparison functions used by `--filter`.

`--sort-locale` ⟨*value*⟩

If the ⟨*value*⟩ is the keyword " `none` ", use letter-sorting for strings. Otherwise ⟨*value*⟩ should be a valid IETF language tag that identifies a locale to sort according to that locale's alphabetical order.

`--sql` ⟨*query statement*⟩

Import data from an SQL database where ⟨*query statement*⟩ is the SELECT query that identifies the required data.

`--sqldb` ⟨*name*⟩

The SQL database name.

`--sqlhost` ⟨*port*⟩

The SQL host.

`--sqlpassword` ⟨*password*⟩

The SQL password (insecure). If omitted, you will be prompted for a password if you try to import data from an SQL database.

`--sqlport` ⟨*port*⟩

The SQL port number.

`--sqlprefix` ⟨*prefix*⟩

The SQL prefix. Currently, only MySQL is supported.

`--sqluser` ⟨*username*⟩

The SQL username.

`--tex-encoding` ⟨*encoding*⟩

Sets the encoding for the TEX files.

`--truncate` ⟨*n*⟩

Truncate the database to the first ⟨*n*⟩ rows.

`--version` (or `-v`)

Prints version information and exits.

`--wipepassword`

Wipe the password from memory as soon as it has been used to connect to an SQL database (default).

`--xls` ⟨*xls file*⟩

Import data from the given Excel `xls` file.

Glossary

CSV	comma-separated values
GUI	graphical user interface
IETF	internet engineering task force
L&F	Look and Feel
SQL	structured query language

Index