PLoT Manual

PLT (scheme@plt-scheme.org)

372
Released December 2007

Copyright notice

Contents

1. Quick Start

1.1 Overview

PLoT (aka PLTplot) provides a basic interface for producing common types of plots such as line and vector field plots as well as an advanced interface for producing customized plot types. Additionally, plots and plot-items are first-class values and can be generated in and passed to other programs.

1.2 Basic Plotting

After loading the correct module using `(require (lib "plot.ss" "plot"))` try `(plot (line (lambda (x) x)))`

Any other function with the contract `number -> number` can be plotted using the same form. To plot multiple items, use the functions `mix` and `mix*` to combine the items to be plotted.

```
(plot (mix (line (lambda (x) (sin x)))
           (line (lambda (x) (cos x)))))
```

The display area and appearance of the plot can be changed by adding parenthesized argument/value pairs after the first argument.

```
(plot (line (lambda (x) (sin x)))
      (x-min -1)
      (x-max 1)
      (title "Sin(x)"))
```

The appearance of each individual plot item can be altered by adding parameter-value pairs after the data.

```
(plot (line (lambda (x) x)
            (color 'green)
            (width 3)))
```

Besides plotting lines from functions in 2d, the plotter can also render a variety of other data in several ways:

- Discrete data, such as

  ```
  (define data
   (list (vector 1 1 2)
         (vector 2 2 2)))
  ```

  can be interpreted in several ways:

  - As points: `(plot (points data))`
  - As Error Data: `(plot (error-bars data))`

- A function of two variables, such as

```
(define 3dfun (lambda (x y) (* (sin x) (sin y))))
```

  can be plotted on a 2d graph
  - Using contours to represent height (z)

```
(plot (contour 3dfun))
```

  - Using color shading

```
(plot (shade 3dfun))
```

  - Using a gradient field

```
(plot (vector-field (gradient 3dfun)))
```

  or in a 3d box
  - Displaying only the top of the surface

```
(plot3d (surface 3dfun))
```

1.3 Curve Fitting

The scheme-plot library uses a Non-Linear Least Squares fit algorithm to fit parametrized functions to given data.

To fit a particular function to a curve:

1. Set up the independent and dependent variable data. The first item in each vector is the independent var, the second is the result. The last item must is the weight of the error—we can leave it as 1 since all the items weigh the same.

```
(define data '(#(0 3 1)
               #(1 5 1)
               #(2 7 1)
               #(3 9 1)
               #(4 11 1)))
```

2. Set up the function to be fitted using fit. This particular function looks like a line. The independent variables must come before the parameters.

```
(define fit-fun
  (lambda (x m b) (+ b (* m x))))
```

3. If possible, come up with some guesses for the values of the parameters. The guesses can be left as one, but each parameter must be named.

4. Do the fit – the details of the function are described in the Curve Fitting section

```
(define fit-result
  (fit fit-fun
       ((m 1) (b 1))
       data))
```

5. View the resulting parameters

```
(fit-result-final-params fit-result) ; will produce ((m 2) (b 3))
```

6. For some visual feedback of the fit result, plot the function with the new parameters. For convenience, the structure that is returned by the fit command has already created the function.

```
(plot (mix (points data)
           (line (fit-result-function fit-result)))
      (y-max 15))
```

A more realistic example can be found in **demos/fit-demo-2.ss**.

1.4 Creating Custom Plots

Defining custom plots is simple: a Plot-item (that is passed to plot or mix) is just a function that acts on a view. Both the 2d and 3d view snip have several drawing functions defined that the plot-item can call in any order. The full details of the view interface can be found in the **plot-extend.ss** section.

For example, if we wanted to create a constructor that creates Plot-items that draw dashed-lines given a number-number function we could do the following: Load the required modules

```
(require (lib "class.ss")
         (lib "etc.ss")
         (lib "plot-extend.ss" "plot"))
```

Set up the constructor

```
(define-plot-type dashed-line
 fun 2dview (x-min x-max) ((samples 100) (segments 20) (color 'red) (width 1))
 (let* ((dash-size (/ (- x-max x-min) segments))
        (x-lists (build-list
                  (/ segments 2)
                  (lambda (index)
                    (x-values
                     (/ samples segments)
                     (+ x-min (* 2 index dash-size))
                     (+ x-min (* (add1 ( * 2 index)) dash-size)))))))
   (send* 2dview
          (set-line-color color)
          (set-line-width width))
   (for-each
    (lambda (dash)
      (send 2dview plot-line
            (map (lambda (x) (vector x (fun x))) dash)))
    x-lists)))
```

Plot a test case

```
(plot (dashed-line (lambda (x) x) (color 'blue)))
```

2. Module: plot.ss

The **plot.ss** module provides the ability to make basic plots, fit curves to data, and some useful miscellaneous functions.

2.1 Plotting

The *plot* and *plot3d* forms generate plots that can be viewed in the DrScheme Interactions window. The functions and data definitions for this module are as follows:

```
Forms:
  (plot 2d-plot-item 2d-plot-option*) -> VIEW
  (plot3d 3d-plot-item 3d-plot-option*) -> VIEW

2d-plot-option is one of:
  (x-min number)
  (x-max number)
  (y-min number)
  (y-max number)
  (x-label string)
  (y-label string)
  (title string)

3d-plot-option is one of:
  2d-plot-option
  (z-label)
  (z-min number)
  (z-max number)
  (alt number)    ; altitude angle, in degrees
  (az number)     ; azimuthal angle, in degrees
```

The 2d and 3d plot-options modify the view in which the graph is drawn. The 3d-plot-options alt and az set the viewing altitude (in degrees) and azimuth (also in degrees) respectively. The rest of the options should be self-explanatory.

```
Data Definitions:

2d-plot-item is one of:
  (points (list-of (vector number number)) point-options*)
  (line [(number -> number) | (number -> (vector number number))] line-options*)
  (error-bars (list-of (vector number number number)) error-bar-options*)
  (vector-field ((vector number number) -> (vector number number)) field-options*)
  (contour (number number -> number) contour-options*)
  (shade (number number -> number) shade-options*)
  (mix 2d-plot-item 2d-plot-item+)
  (custom (2d-view\% -> void))
```

```
3d-plot-item is one of:
  (surface (number number -> number) surface-options*)
  (mix 3d-plot-item 3d-plot-item+)
  (custom (3d-view\% -> void))

note: all of the options appear as
  option-name : enumeration of values with default enclosed in []
or
  option-name : type [default]

color is one of:
  'white 'black 'yellow 'green 'aqua 'pink
  'wheat 'grey 'brown 'blue 'violet 'cyan
  'turquoise 'magenta 'salmon 'red

point-options are:
  sym       : ['square], 'circle, 'odot, 'bullet
  color     : color ['black]

line-options are:
  samples   : number [150]
  width     : number [1]
  color     : color ['red]
  mode      : ['standard], 'parametric
  mapping   : ['cartesian], 'polar
  t-min     : number [-5]
  t-max     : number [5]

error-bar-options are:
  color     : color ['red]

field-options are:
  color     : color ['red]
  width     : number [1]
  style     : ['scaled],'normalized,'read

contour-options are:
  samples   : number [50]
  color     : color ['black]
  width     : number [1]
  levels    : number U (list-of number) [10]

shade-options are:
  samples   : number [50]
  levels    : number [10]

surface-options are:
  samples   : number [50]
  color     : color ['black]
  width     : number [1]
```

The 2d and 3d plot-items can be created in several ways. The first is by using the built-in constructors with your own data.

points will draw points on a graph given a list of vectors specifying their location. Sym specifies the appearance of the points.

line will draw a line specified in either functional, ie. y=f(x), or parametric mode, x,y = f(t). If the function is parametric, the line-option *mode* must be set to *parametric*. *t-min* and *t-max* set the parameter when in parametric mode. *mapping* can be set to 'radial.

error-bars will draw error bars given a list of vectors. The vector specifies the center of the error bar (x,y) as the first two elements, and its magnitude as the third.

vector-field will draw a vector field from a vector valued function. Styles are either *real*, *scaled*, or *normalized*.

Both *shade* and *contour* will render 3d functions on a 2d graph using colored shades and contours (respectively) to represent the value of the function at that position. *contour* will let you choose the levels explicitly if desired, by setting the *levels* option to a list of contour levels to be plotted.

surface plots a 3d surface in a 3d box, showing only the *top* of the surface.

2.2 Curve Fitting

PLTPlot uses the standard Non-Linear Least Squares fit algorithm for curve fitting. The code that implements the algorithm is public domain, and is used by the gnuplot package.

```
Data:

  a fit-result is
    (fit (number*-> number) parameter-guess-list  data)

    parameter-guess-list is a set of name-value pairs enclosed in (..)

    data is
      (list-of (vector number number number))
    | (list-of (vector number number number number))

Functions:

  fit-result-function : fit-result -> procedure
  fit-result-final-params : fit-result -> guess-list
```

The *fit* form attempts to fit a *fittable-function* to the data that is given. The *guess-list*should be set of parameters and values. The more accurate your initial guesses are, the more likely the fit is to succeed. If there are no good values for the guesses, leave them as 1.

fit-result-final-params returns an associative list of the parameters specified in fit and their values. Note that the values may not be correct if the fit failed to converge. For a visual test use *fit-result-function* to get the function with the parameters in place and plot it along with the original data.

2.3 Misc Functions

The plot library comes with a few useful miscellaneous functions:

```
derivative      : (number -> number) [h .000001] -> (number -> number)
gradient : (number number -> number) [h .000001] -> (vector -> vector)
make-vec : (number number -> number) (number number -> number) -> (vector -> vector)
```

`derivative` creates a function that evaluates the numeric derivative of the given single-variable function using the definition. `h` is the divisor used in the calculation.

`gradient` creates a vector-valued function that is the gradient of the given function. `h` represents the numeric divisor, as with the derivative function.

`make-vec` creates one vector valued function from two parts.

3. Module: plot-extend.ss

plot-extend.ss allows you to create your own constructors, further customize the appearance of the plot windows, and in general extend the package.

3.1 Functions

The following are defined in **plot-extend.ss** as utilities for extensions to PLTPlot.

- *define-plot-type*
- *2d-view\%*
- *3d-view\%*
- *sample-size: number number number -> number*
  Given samples, x-min and x-max, returns the size of each sample.
- *scale-vectors* : (listof *vector*) *number number* -> (listof *vector*)
  Scales vectors, causing them to fit in their boxes. First argument is vectors, second is x-sample-size, and third is y-sample-size.
- *x-values: number number number -> listof-number*
  Given samples, x-min and x-max, returns a list of the x's spread across that range.
- *normalize-vector: vector number number -> vector*
  Given a vector, the x-sample-size, and the y-sample-size, normalizes that vector.
- *normalize-vectors:* (listof *vector*) *number number -> vector*
  Maps *normalize-vector* across the vectors.
- *make-column: number* (listof *number*) -> (listof *vector*)
  Given an x and a list of y's, produces a list of points pairing the x with each of the y's.
- *xy-list: number number number number number* -> (listof *vector*)
  Makes a list of all the positions on the graph.
- *zgrid* : (*number number -> number*) (listof *number*) (listof *number*) -> (listof (listof *number*))
  Given a function that consumes x and y's, a list of x's, and a list of y's, produces a list of z column values.

3.2 2d-view%

Provides an interface to drawing 2dplots. Some methods call low-level functions while others are emulated in scheme.

- *set-labels* : *string string string* -> void
  Sets x, y and title labels
- *plot-vector* : *vector vector* -> void
  Plots a single vector. First argument is the head, second is the tail.
- *plot-vectors* : (listof (list *vector vector*)) -> void
  Plots a list of vectors. Each vector is a list of two scheme *vector*s.
- *plot-points* : (listof *vector*) *number* ->*void*
  Plots points using a specified character. ** provide character map **
- *plot-line* : (listof (*vector number number*)) -> void
  Plots a line given a set of points. Each point is represented by a *vector*.
- *plot-contours* : (listof (*lisftof number*)) (listof *number*) (listof *number*) (listof *number*) ->*void*
  Plots a grid representing a 3d function using contours to distinguish levels. Args are grid, xvalues yvalues and levels to plot.
- *plot-shades* : (listof (*lisftof number*)) (listof *number*) (listof *number*) (listof *number*) ->*void*
  Plots a grid representing a 3d function using shades to represent height (z).

3.3 3d-view%

Provides an interface to drawing 3d plots.

- *plot-surface* : (listof (*lisftof number*)) (listof *number*) (listof *number*) ->*void*
  Plots a grid representing a 3d function in a 3D box, showing only the top of the surface.
- *plot-line* : (listof *number*) (listof *number*) (listof *number*) -> void
  Plots a line in 3d space. The arguments are lists of x,y and z coordinates respectively.
- *get-z-min* : -> *number*
  Returns the minimum plottable Z coordinate.
- *get-z-max* : -> *number*
  Returns the maximum plottable Z coordinate.
- *get-alt* : -> *number*
  Returns the altitude (in degrees) from which the 3d box is viewed.
- *get-az* : -> *number*
  Returns the azimuthal angle.

3.4 define-plot-type

Macro used to create new constructors. It is easiest to explain with an example, so here is an implementation of a simple line constructor:

```
(define-plot-type line
  func 2dplotview (x-min x-max) ((samples 150) (color 'red) (width 1))
  (send* 2dplotview
```

```
(set-line-color color) (set-line-width width)
(plot-line (map (lambda (x) (vector x (func x)))
                (x-values samples x-min x-max)))))
```

- The first keyword after then name of the new plot type, is used to refer to the data that will be rendered. In this case, we will are calling our data *func*. For example, in the execution of `(plot (line (lambda (x) x))) func` would refer to the identity function.
- *2dplotview* refers to the name of the view object that the Plot-item will be applied to by *plot*
- The *x-min* and *x-max* are fields in the *2d-view\%* object. They will be bound to the values of those fields before the execution of the body, assuming the object has the methods *get-x-min* and *get-x-max*. This entire expression can be omitted if none of the fields are necessary (such as for plotting discrete data points).
- The last set of parenthesized expressions sets keywords based arguments and their default values for the constructor. To over-ride values the user needs to provide an associative list with the desired values. Ex: `(line (lambda (x) x) '((color blue)))`

License

GNU Library General Public License

Version 2, June 1991

675 Mass Ave, Cambridge, MA 02139, USA

[This is the first released version of the library GPL. It is numbered 2 because it goes with version 2 of the ordinary GPL.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users.

This license, the Library General Public License, applies to some specially designated Free Software Foundation software, and to any other libraries whose authors decide to use it. You can use it for your libraries, too.

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

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

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link a program with the library, you must provide complete object files to the recipients so that they can relink them with the library, after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

Our method of protecting your rights has two steps: (1) copyright the library, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the library.

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

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that companies distributing free software will individually obtain patent licenses, thus in effect transforming the program into proprietary software. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License, which was designed for utility programs. This license, the GNU Library General Public License, applies to certain designated libraries. This license is quite different from the ordinary one; be sure to read it in full, and don't assume that anything in it is the same as in the ordinary license.

The reason we have a separate public license for some libraries is that they blur the distinction we usually make between modifying or adding to a program and simply using it. Linking a program with a library, without changing the library, is in some sense simply using the library, and is analogous to running a utility program or application program. However, in a textual and legal sense, the linked executable is a combined work, a derivative of the original library, and the ordinary General Public License treats it as such.

Because of this blurred distinction, using the ordinary General Public License for libraries did not effectively promote software sharing, because most developers did not use the libraries. We concluded that weaker conditions might promote sharing better.

However, unrestricted linking of non-free programs would deprive the users of those programs of all benefit from the free status of the libraries themselves. This Library General Public License is intended to permit developers of non-free programs to use free libraries, while preserving your freedom as a user of such programs to change the free libraries that are incorporated in them. (We have not seen how to achieve this as regards changes in header files, but we have achieved it as regards changes in the actual functions of the Library.) The hope is that this will lead to faster development of free libraries.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, while the latter only works together with the library.

Note that it is possible for a library to be covered by the ordinary General Public License rather than by this special one.

GNU LIBRARY GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Library General Public License (also called "this License"). Each licensee is addressed as "you".

   A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

   The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

   "Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

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

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

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
   a) The modified work must itself be a software library.
   b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
   c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
   d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.
      (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

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

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

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

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

   Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

   This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

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

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

   However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

   When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

   If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

   Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also compile or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

   You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

   a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)
   b) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.
   c) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.
   d) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

   For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

   a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.
   b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

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

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

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

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

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

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

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

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

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

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

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

NO WARRANTY

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

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

END OF TERMS AND CONDITIONS

Index