

Copyright ©2006 Alex Gutteridge

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Contents

| 1 | Overview 1.1 About RSRuby | |
|---|---|----------------|
| 2 | Installation and Getting Started 2.1 Requirements | |
| 3 | Accessing R From Ruby 3.1 Looking Up R Objects From Ruby 3.2 The RObj Class | |
| 4 | Conversion System 1 4.1 R to Ruby 2 4.1.1 Modes 3 4.2 Ruby to R 3 | 13 |
| 5 | 5.0.1 Enhanced RObj | 17 17 19 |
| 6 | Input/Output | 21 |
| 7 | Known Issues and To Do 2 7.1 Known Issues 5 7.2 To Do 5 7.2.1 Bugs 5 7.2.2 Features 5 | 23 23 |
| 8 | Acknowledgements | 25 |
| 9 | GNU Free Documentation License | 27 |

4 CONTENTS

Overview

1.1 About RSRuby

RSRuby is a Ruby interface to the R programming language. R has an extensive collection of libraries and functions for statistical and scientific computation which are currently unavailable in native Ruby. RSRuby allows a Ruby programmer to take advantage of these resources from within a Ruby script.

RSRuby is inspired by, and built from, two other projects: RSPerl (http://www.omegahat.org/RSPerl/), which provides an R interface for Perl, and RPy (http://rpy.sourceforge.net/), an interface for Python. The current version of RSRuby is built almost exclusively from RPy derived code, but previous versions were built from RSPerl code and hence the name has been kept.

The current goal of the project is to implement the full RPy feature set (this is approximately 80% complete as of version 0.5). Future versions may move towards providing a more Rubyesque interface and feature set. With this in mind the current goals of RSRuby are very similar to those of RPy:

Robustness: The library itself should be as stable as possible. RSRuby should never

segfault or hang.

Transparency: Code written using RSRuby should be readable for both R and Ruby

programmers. It should also be relatively straightforward for users to

translate R scripts into RSRuby.

Real World: RSRuby should be fast enough and support enough libraries and R features

to allow it to be used for any task where R is currently used.

1.2 Contact info and Contributing

RSRuby is hosted at Rubyforge (http://rubyforge.org/projects/rsruby/). Bug reports, feature requests, cries for help, patches and so on should be made using the features on that site. We encourage all bug reports to include Test::Unit compatible tests demonstrating the bug. Please see the test directory for examples. RSRuby is currently developed by Alex Gutteridge (alexg@kuicr.kyoto-u.ac.jp).

Installation and Getting Started

2.1 Requirements

Obviously Ruby (http://www.ruby-lang.org/) is required before installing RSRuby. The current version of RSRuby has been tested with Ruby version 1.8.4 only, though earlier versions in the 1.8.x family may work.

A working R installation is also required. R must have been installed and built with the --enable-R-shlib option enabled to provide the R shared library that RSRuby interfaces with. The current version of RSRuby has been tested on R version 2.2.1, but earlier versions may work.

2.2 Installation

Installation requires some correct configuring of your R setup and then a standard Ruby extension install via setup.rb or rubygems.

1. First, ensure that the R_HOME environment variable is set correctly. On my Ubuntu linux machine this is:

```
R_HOME=/usr/lib/R
```

While on OS X it is:

```
R_HOME=/Library/Frameworks/R.framework/Resources
```

2. Compile and install the Ruby library using setup.rb as shown. You need to supply the location of your R installation for the libR shared library. This will usually be the same as R_HOME.

```
cd rsruby
ruby setup.rb config -- --with-R-dir=/usr/lib/R
ruby setup.rb setup
sudo ruby setup.rb install
```

If you prefer to use rubygems then you just need to supply the location of the R library to gem install:

```
gem install -- --with-R-dir=/usr/lib/R
```

If RSRuby does not compile you may need to configure the path to the R library. Any one of the following should be sufficient (taken from the RPy documentation):

- make a link to R_HOME/bin/libR.so in /usr/local/lib or /usr/lib, then run ldconfig,
- or, put the following line in your .bashrc (or equivalent):

```
export LD_LIBRARY_PATH=\$LD_LIBRARY_PATH:R_HOME/bin
```

• or, edit the file /etc/ld.so.conf and add the following line:

```
R_HOME/bin
```

and then, run ldconfig.

3. Test it:

```
ruby setup.rb test
```

RSRuby should pass all tests.

You can avoid needing root in the install step by providing setup.rb with a suitable install directory (such as home). Please run ruby setup.rb --help for more details.

2.3 Getting Started: A Small Example

RSRuby works fine with irb. A sample session is shown:

```
\$irb -rrsruby
irb> r = RSRuby.instance
=> #<RSRuby:0xb7cdec04>
```

require-ing rsruby loads the module, but unlike RPy it does not start the R interpreter. Calling RSRuby.instance starts the R interpreter, a reference to which is returned and stored in object r here. Since only one R interpreter can be running at a time, RSRuby uses the standard library Singleton module. This replaces RSRuby.new with RSRuby.instance. further calls to RSRuby.instance return the original interpreter object.

Calls to R functions are made via the R interpreter object. In the example below the R function wilcox.test is called with two Ruby Arrays as arguments. These Arrays are converted to R by RSRuby. The list returned from the R method is converted to a Ruby Hash. Details of the conversion procedure can be found in Chapter 4.

```
irb> r.wilcox_test([1,2,3],[4,5,6])
=> {"p.value"=>0.1, "null.value"=>{"mu"=>0.0},
    "data.name"=>"c(1, 2, 3) and c(4, 5, 6)",
    "method"=>"Wilcoxon rank sum test",
    "alternative"=>"two.sided", "parameter"=>nil,
    "statistic"=>{"W"=>0.0}}
```

Accessing R From Ruby

3.1 Looking Up R Objects From Ruby

R functions and variables are both accessed from Ruby in the same way. Unlike in Ruby, R variables and functions cannot share the same name, so there is no possibility of accessing a function when you meant to access a variable (except by user error).

Similarly to RPy, there are two methods for accessing R objects from Ruby. However, unlike RPy, the two methods are not exactly equivalent so some care should be taken.

The first method of retrieving an R object is to call a method on the RSRuby interpreter object:

```
irb> r = RSRuby.instance
=> #<RSRuby:0xb7d30c04>
irb> r.seq
=> #<R0bj:0xb7d2bc2c>
irb> r.as_data_frame
=> #<R0bj:0xb7d29c88>
irb> r.print_
=> #<R0bj:0xb7d27f8c>
```

In the case of seq, as_data_frame and print_ the following R functions are returned: seq, as.data.frame and print. These functions are encapsulated in Ruby as an object of class RObj, about which we will learn more later in section 3.2. The name conversion is required to make the strings valid Ruby method names. The rules of the name conversion are identical to RPy as shown in Table 3.1.

| Ruby name | R name |
|---|--------------|
| Underscore ('_') | Dot ('.') |
| Double underscore ('') | Arrow ('<-') |
| Final underscore (preceded by a letter) | Is removed |

Table 3.1: Name conversion table.

As mentioned above, the same syntax is used for retrieving R variables, as shown below:

```
irb> r.foo
RException: Error in get(x, envir, mode, inherits) :
```

```
variable "foo" was not found

from /ruby/site_ruby/1.8/rsruby.rb:351:in 'lcall'
from /ruby/site_ruby/1.8/rsruby.rb:351:in '__getitem__'
from /ruby/site_ruby/1.8/rsruby.rb:207:in 'method_missing'
from (irb):5
irb> r.assign('foo',42)
=> 42
irb> r.foo
=> 42
```

In the first case the R object foo is requested by RSRuby and an RException thrown when it is not found. After calling assign the foo variable exists in R and so can be found by RSRuby.

Note that when a method is called on the R interpreter with out any arguments, a Ruby object representing the R object of the same name is returned. However, when called with arguments, the object requested is assumed to be an RObj (or subclass thereof) and is called with the arguments given. The conversion of the arguments from Ruby to R is covered in Chapter 4 and the various calling semantics in section 3.2.1.

The second way to access R objects is via a Hash or Array style interface using '[]'. This method may seem redundant given the method call syntax given above, but it is the only way to access R functions such as '[[':

```
irb> r = RSRuby.instance
=> #<RSRuby:0xb7b32224>
irb> foo = r.as_data_frame(1)
=> {"1"=>1}
irb> r['[['].call(foo,1)
=> 1
```

Note that the '[]' syntax does not call the returned object automatically. Instead, in this example, the RObj returned by '[]' must have the call function called on it. This is discussed in section 3.2.1 below.

3.2 The RObj Class

R functions and other objects not handled by the conversion system are returned as RObj objects. The RObj class defines four methods:

as_ruby: as_ruby forces the conversion of the RObj into Ruby according to the current conversion system. If conversion cannot be done then the same RObj is returned.

call(args): call treats the RObj as a function and attempts to call it with the arguments given. Internally call translates its arguments and delegates to lcall.

lcall(args): lcall is the lowest level calling function (defined in C). It is described below.

autoconvert: Sets the conversion mode for this object.

3.2.1 Calling R Functions

The rules for calling R functions are similar to RPy but differ in a few details. Specifically, Ruby does not directly support named arguments, while R allows the user to mix positional and named arguments. As a workaround to this, if call is called on an RObj and the last argument is a Hash then the Hash is treated as a collection of named arguments for the function. Other arguments are converted to Ruby as appropriate for the current conversion mode. Note that if an R function requires a single R list (usually represented as Ruby Hash) as their only argument, the lcall form given below must be used to prevent the Hash being converted to a set of named arguments.

The most basic method of function calling is to use lcall. In this function arguments should be of the form of an Array of two member Arrays. The first element in each Array is the variable name and can be left as an empty string if needed. lcall allows R functions that require both named and ordered arguments. Some examples to summarise:

First of all, calling a function without named arguments (sum). All of the following are equivalent:

```
irb> r.sum(1,2,3)
=> 6
irb> r.sum.call(1,2,3)
=> 6
irb> r.sum.lcall([['',1],['',2],['',3]])
=> 6
```

Second, calling a function with named arguments (seq):

```
irb> r.seq(:length => 10, :from => -5, :by => 1)
=> [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]
irb> r.seq.call(:length => 10, :from => -5, :by => 1)
=> [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]
irb> r.seq.lcall([['length',10],['from',-5],['by',1]])
=> [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]
```

Lastly, mixing named and non-named arguments:

```
irb> r.paste('foo','bar',{:sep => "baz"})
=> "foobazbar"
irb> r.paste.call('foo','bar',{:sep => "baz"})
=> "foobazbar"
irb> r.paste.lcall([['','foo'],['','bar'],['sep','baz']])
=> "foobazbar"
```

Conversion System

4.1 R to Ruby

RSRuby has four different conversion modes. Each mode can be applied globally (in which case it effects all functions) and locally to single functions. The local mode. The conversion system can be left in the default mode most of the time, but if complex conversion is required it is also customisable in a number of ways.

4.1.1 Modes

The different modes are detailed below each is specified by a constant defined in the RSRuby namespace (e.g. RSRuby::PROC_CONVERSION). The constants are as follows:

- PROC_CONVERSION
- CLASS_CONVERSION
- BASIC_CONVERSION
- VECTOR_CONVERSION
- NO_CONVERSION
- NO_DEFAULT

To manipulate the conversion modes the following functions are available from the RSRuby class:

get_default_mode(): Returns the current default (global) mode.

The results of calls to RObj objects are then converted to Ruby according to the following rules:

- 1. Unless the default mode is set to NO_DEFAULT then the current default mode is used.
- 2. If the current default mode is set to NO_DEFAULT then the RObj's local mode is used.
- 3. If no appropriate conversion can be performed in the current default mode then the object falls through to the next mode (the order is PROC_CONVERSION, CLASS_CONVERSION, BASIC_CONVERSION, VECTOR_CONVERSION, NO_CONVERSION. NO_CONVERSION always succeeds and returns an RObj representing the returned value.

Proc Conversion

This is a user customisable mode which converts an R object according to functions placed in the proc_table Hash stored in the RSRuby object. The keys of this Hash are Proc objects which take a single argument. The single argument is the RObj representation of the object to be transformed. Note that because Ruby Hashes are not ordered, the order with which the functions are tested cannot be defined. A new implementation using an ordered hash (or Array of tuples) would be preferred.

Once a Proc (hash key) has been found that returns true the corresponding value (which is also a Proc accepting one argument) is used to convert the RObj. The value returned by this Proc should return the required Ruby representation.

An example may be clearer:

```
irb> r = RSRuby.instance
=> #<RSRuby:0xb7add6c4>
irb> check_str = lambda{|x| RSRuby.instance.is_character(x)}
=> #<Proc:0xb7ad6180@(irb):2>
irb> reverse = lambda{|x| x.to_ruby.reverse}
=> #<Proc:0xb7acbb2c@(irb):3>
irb> r.proc_table[check_str] = reverse
=> #<Proc:0xb7acbb2c@(irb):3>
irb> RSRuby.set_default_mode(RSRuby::PROC_CONVERSION)
=> 4
irb> r.paste("hello","world")
=> "dlrow olleh"
irb> r.sum(1,2,3)
=> 6
```

Here we set up a Proc, check_str, to check whether a string is returned by the R function. If it does then the reverse Proc is called. This simply converts the RObj to a Ruby object, in this case a String, using to_ruby and reverses it. The conversion is demonstrated using paste, which returns a String (and so is reversed) and sum which returns an Integer and so is unaffected.

As with RPy, within the conversion routine the conversion mode is switched to basic to prevent infinite recursion. Since each key is tested this mode can be inefficient if the proc_table becomes large. It should also be noted that by using a single key that always returns true the RSRuby conversion system can be completely bypassed and a custom conversion system plugged in.

Class Conversion

The CLASS_CONVERSION mode is similar to the PROC_CONVERSION mode in that a built in Hash is consulted for an appropriate conversion routine for each object. In this case the class_table Hash is consulted. The keys of this Hash are Strings, or Arrays of Strings. The values of the Hash are conversion Procs as used in the PROC_CONVERSION mode.

Each object returned by R is tested against the keys of class_table. In order for an object to match, one of the following must be satisfied:

- The R attribute class is a string and it is found in class_table.
- The R attribute class is a vector of strings and it is found in class_table.

4.1. R TO RUBY

• The R attribute class is a tuple of strings and one of the tuple's elements is found in class_table.

In the example below, all R objects of class 'data.frame' are returned as the Ruby Integer 5:

```
irb> r = RSRuby.instance
=> #<RSRuby:0xb7ae7224>
irb> RSRuby.set_default_mode(RSRuby::CLASS_CONVERSION)
=> 3
irb> conversion = Proc.new{|x| 5}
=> #<Proc:0xb7adde04@(irb):3>
irb> r.class_table['data.frame'] = conversion
=> #<Proc:0xb7adde04@(irb):3>
irb> r.as_data_frame([1,2,3])
=> 5
```

Basic Conversion

BASIC_CONVERSION mode attempts to convert the R object into a basic (by which I mean standard library) Ruby type. R types are generally not perfectly matched by basic Ruby types so some data might be lost in this conversion. If this is not desired by the user then the custom conversion modes (see above) should be used. Table 4.1 shows the conversion rules:

| ruies: | | |
|----------|----------------------------|--------|
| R object | Ruby Class | Notes |
| NULL | nil | |
| Logical | TrueClass or FalseClass | a,b |
| Integer | Integer (Fixnum or Bignum) | a,b |
| Real | Float | a, c |
| Complex | Complex | a |
| String | String | a |
| Vector | Array or Hash | a,d |
| List | Array or Hash | a,d |
| Array | Array | d |
| Other | Fails | |
| | | |

^a In R there are no true scalar types. All values are vectors, with scalars represented as vectors with length one. This can be confusing for Ruby programmers and so vectors of length one are converted to Ruby scalars while vectors of length more than one are returned as Arrays.

Table 4.1: Basic mode conversions between R and Ruby in RSRuby.

Vector Conversion

VECTOR_CONVERSION mode is exactly the same as BASIC_MODE *except* that it always returns a Ruby Hash or Array no matter what the length of the returned R vector.

^b In R there is 'NA' (not applicable) which is converted to Ruby as the lowest possible Fixnum in Ruby (which is system dependant).

^c The IEEE float values NaN and Inf are also converted to and from R and Ruby.

^d Vectors and lists in R can have a name attribute. If this attribute is set then the vector/list is converted to a Ruby Hash. When there is no name attribute a Ruby Array is returned. Note that Hashes do not retain any order information.

No Conversion

The final conversion mode is NO_CONVERSION if all attempts at previous modes fails then they fall through to this mode which returns an RObj (a reference to the R object). This mode should always succeed.

4.2 Ruby to R

Converting from Ruby to R is more straightforward. When a Ruby object is passed to an R function, the following steps are attempted until one succeeds:

- 1. If the Ruby object defines a as_r method then the result of that method is converted to R. If you have custom classes that you wish to be able to pass to R then this is the method you must define.
- 2. If the Ruby object is of class RObj then it is converted into the R object it represents.
- 3. The Ruby object is converted according to Table 4.1.
- 4. An ArgumentError exception is thrown giving the message 'Unsupported object passed to R'.

Extending RSRuby and Further Examples

For most simple applications the basic conversion mode is often sufficient, however if you use R objects that go beyond the basic types you will need to start playing with the PROC_CONVERSION and CLASS_CONVERSION modes. In tandem with some custom Ruby classes these can make quite powerful systems:

5.0.1 Enhanced RObj

This example shows an extended RObj class similar to that demonstrated in the RPy manual. It uses method_missing to replace attribute lookup on the R object. It also demonstrates a method of replicating the string output given by R. This code is included in the RSRuby distribution and can be activated by require-ing rsruby/erobj and setting the proc_table appropriately as shown:

```
class ERObj
  @@x = 1
  def initialize(robj)
    @robj = robj
    @r = RSRuby.instance
  end
```

The \mathtt{ERObj} initialization method simply stores the wrapped \mathtt{RObj} and the RSRuby interpreter.

```
def as_r
    @robj.as_r
end
def lcall(args)
    @robj.lcall(args)
end
```

The as_r and lcall methods simply delegate to the same methods in the wrapped underlying RObj.

```
def to_s
  @@x += 1
```

```
mode = RSRuby.get_default_mode
RSRuby.set_default_mode(RSRuby::NO_CONVERSION)
a = @r.textConnection("tmpobj#{@@x}",'w')
RSRuby.set_default_mode(RSRuby::BASIC_CONVERSION)
@r.sink(:file => a, :type => 'output')
@r.print_(@robj)
@r.sink.call()
@r.close_connection(a)
str = @r["tmpobj#{@@x}"].join("\n")
RSRuby.set_default_mode(mode)
return str
end
```

The to_s method makes the R interpreter print to the tmpobj variable using the textConnection and sink functions. This is then retrieved and returned as the string representation of the object for Ruby.

```
def method_missing(attr)
  mode = RSRuby.get_default_mode
  RSRuby.set_default_mode(RSRuby::BASIC_CONVERSION)
  e = @r['\$'].call(@robj,attr.to_s)
  RSRuby.set_default_mode(mode)
  return e
  end
end
```

The method_missing function returns the attribute with the same name as the missing method from the wrapped RObj.

To use the ERObj class we can set the proc_table to return a new ERObj with every conversion.

```
irb> r = RSRuby.instance
=> #<RSRuby:0xb7baf320>
irb> r.proc_table[lambda{|x| true}] = lambda{|x| ERObj.new(x)}
=> #<Proc:0xb7ba68ec@(irb):2>
irb> RSRuby.set_default_mode(RSRuby::PROC_CONVERSION)
=> 4
```

To test the returned class we use the R t.test function, which returns an R list. Note the string representation of the returned object which matches the string form given by R.

```
irb> e = r.t_test([1,2,3,4,5,6])
=> #<ERObj:Oxb7b9d918>
irb> puts e

One Sample t-test

data: c(1, 2, 3, 4, 5, 6)
t = 4.5826, df = 5, p-value = 0.005934
```

5.0.2 ArrayFields

The default conversion mode converts R lists into Ruby Hashes. This changes the semantics of the returned object because R lists retain order information while Ruby Hashes are unordered. To fix this we can plug a new method in to return Arrays extended by the ArrayFields gem:

```
require 'rubygems'
require_gem 'arrayfields'
require 'rsruby'

test_proc = lambda{|x| !(RSRuby.instance.attr(x,'names').nil?)}
```

This Proc tests the given object to see if the names attribute is set. If so, the following Proc is used to convert the object.

```
conv_proc = lambda{|x|
  hash = x.to_ruby
  array = []
  array.fields = RSRuby.instance.attr(x,'names')
  RSRuby.instance.attr(x,'names').each{|f| array[f] = hash[f]}
  return array
}
```

We then setup the proc_table with these procs and test using the R t.test function.

```
r = RSRuby.instance
r.t_test.autoconvert(RSRuby::PROC_CONVERSION)
r.proc_table[test_proc] = conv_proc

r.t_test([1,2,3]).each_pair{|f,v| puts "#{f} - #{v}"}
```

Note that the point here is that the list returned by t.test is converted to an Array (with the Arrayfields extensions), which retains order information unlike the Hash it would be converted to otherwise.

5.0.3 DataFrames

DataFrames are useful objects in R programming

Chapter 6 Input/Output

RPy provides a number of utility functions

Known Issues and To Do

7.1 Known Issues

- The conversion does not work correctly with certain R libraries notably Bioconductor.
- Plotting vectors requires manual labelling.
- Ruby functions/procs cannot be passed into R

7.2 To Do

7.2.1 Bugs

7.2.2 Features

Chapter 8 Acknowledgements

GNU Free Documentation License

Version 1.2, November 2002 Copyright ©2000,2001,2002 Free Software Foundation, Inc.

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document 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.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation 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. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.