创建 R 的包

第一章　创建 R 的包

包（Packages）提供了一种机制在需要的时候来加载可供选择的代码和附加文档。在 R 的标准发行版本中提供了一些包。

在本书中，我们假设你知道命令”library()”及其”lib.loc”参数，并且还假设安装包的一些基础知识。另外，在继续阅读前请参考 R 的帮助文档：

```
?library
?INSTALL
```

假设你已经安装好了需要用到的软件，在手册《R Installation and Administration》中指明了需要哪些工具。在 Unix 及其相似的操作系统下，大部分工具是默认提供的，但在 Windows 和 MacOS X 中，就需要手动安装。

创建的包必需通过 R 的命令 INSTALL 来安装。更详细的信息参考《R Installation and Administration》的“Add-on packages”部分。R 也支持其它类型的扩展，请考 1.11 节

1.1 包的结构

一个包是由一些子目录构成的一个目录，它包括一个文件”DESCRIPTION”和子目录 ”R”, ”data”, ”demo”, ”exec”, ”inst”, ”man”, “po”, “src”和”tests”(其中一些目录可以不要)。包的目录下还包括文件”INDEX”, “NAMESPACE”, “configure”, “cleanup”, “LICENSE”, “LICENCE”, “COPYING” 和”NEWS”。其它的像文件”README”或”ChangeLog” 将被 R 忽略掉，但它们可能对用户很有用。

文件”DESCRIPTION”和”INDEX”将在这节说明，文件”NAMESPACE”将在 1.6 节说明。

可选文件”configure”和”cleanup”是 Unix 及其类似的操作系统上的脚本文件（Bourne shell），它们将在安装前和（如果指定了参数“--clean”）安装后执行。参考 1.2

可 选文件”LICENSE”/”LICENCE”或”COPYING”(前者更常用)包括包的许可的拷贝，即 GUN public license。虽然，你可以自由地在你发行的包中加入授权文件，但请不要在你的包中安排安装另一个版本的 GNU "COPYING"或"COPYING.LIB"文件，但可以参考 http://www.r-project.org/Licenses/ 或在 R 的安装目录("share\license")下的 GUN public license 的拷贝。

作为惯例，GNU 项目中的文件”NEWS”和”ChangeLog”，请参考 http://www.gnu.org/prep/standards/standards.html#Documentation.

包的目录名称应当和包的名称一样。因为一些文件系统（像 Windows 的）对大小写不敏感， 为了保持可移植性，强烈建议在这种情况下不要用大小写来区分不同的包。例如你已经有了一个名为”foo”的包，就不要再创建一个名为”Foo”的包。

确 保文件名能在不同的文件系统和不同的操作系统上通用，ASCII 控制字符，像‘”’, ‘*’, ‘:’, ‘/’, ‘<’, ‘>’, ‘?’, ‘\’和‘|’不允许出现在文件名中。另外，含有”con”,”prn”,”aux”,”clock$”,

“nul”, “com1”到”com9”，和”lpt1”到”lpt9”的文件名在转换成小写或者去掉扩展名后，它们可能成为扩展名（如 “lpt5.foo.bar”），这也是不允许的。同样，也不要在同一个文件夹下用大小写来区别不同的文件名（见上段）。除此之外，”.Rd”的文件将用 在 URLs 中，帮必需为 ASCII 码并且不能含有%。为了达到最大的可移植性，文件名应当仅包括 ASCII 字符不要包括已排除的（即 A-Za- z0-9._!#$%&+,;=@^(){}’[]）我们排除了空格，因为在很多就用程序中不受空格在文件名中）：非英文字符不能保证在所有地区 都支持。在实际就用中，最好避免使用壳元字符（）{}’[]$。

一个包如果可能，尽量不要包含二进制的可执行文件：它们是不可移植的，并且如果它们没有很好的构造，也会有安全上的风险。如果不将它们列在包或文件集的最高级文件“BinaryFiles”中，R 的命令 check 将发出警告。
R 的函数 package.skeleton 能够帮助建立一个新的包的结构：更多内容请参见帮助。

1.1.1 文件“DESCRIPTION”
文件”DESCRIPTION”包括包的如下基本信息：
Package: pkgname
Version: 0.5-1
Date: 2004-01-01
Title: My First Collection of Functions
Author: Joe Developer <Joe.Developer@some.domain.net>, with
contributions from A. User <A.User@whereever.net>.
Maintainer: Joe Developer <Joe.Developer@some.domain.net>
Depends: R (>= 1.8.0), nlme
Suggests: MASS
Description: A short (one paragraph) description of what
the package does and why it may be useful.
License: GPL (>= 2)
URL: http://www.r-project.org, http://www.another.url

续 上一行的文字（比如，用一行写不完的描述性文字）是以一个空格或 tab 开始的。
字段‘Package’, ‘Version’, ‘License’, ‘Description’, ‘Title’, ‘Author’, 和‘Maintainer’是必须有的，其它的字段(‘Date’, ‘Depends’, ‘URL’, . . .)是可选的。
为了达到最大化的可移植性，文件”DESCRIPTION”应当完全用 ASCII 字符书写。
字 段”Package”和”Version”分别指出包的名称和版本。名称应当由字母、数字和点组成，并且以字母开头。版本号应当是由”.”或”-”分开的 至少有 2 个（通常是 3 个）非负整数的序列。上面的例子所示的就是其规范形式，并且像“0.01”或“0.01.0”之类的版本号会被处理成 “0.1-0”。（翻译包允许形如“Translation-11”的名字。）
字段“License”应当按以下的标准形式指明包的授权(license)。竖杠（“|”）表示任选其中的一项。单个的授权说明文字必须是下列之一：

- 标准的短授权文字中的一个：GPL-2 GPL-3 LGPL-2 LGPL-2.1 LGPL-3 AGPL-3 Artistic-1.0 Artistic-2.0

就像 http://www.r-project.org/Licenses/或者包含在 R 的源或安装目录下的子目录

“share/licenses”中那样。

- 自由或开放软件授权的缩写的名称，就像包含在 R 的源或安装目录下子目录下的数据库文件‘share/licenses/license.db’中的授权。 它可能（在版本化的授权中）还伴随一个“(op v)”形式的授权版本的限制，其中 op 是‘<’, ‘<=’, ‘>’, ‘>=’, ‘==’,或‘!=’中的一个比较运算符，v 是版本号（用‘.’隔开的非负整数），授权版本限制还可以通过“,”来进行组合（参看下面的例子）。对于版本 化的授权（versioned licenses），也可以在版本号后指明授权的名称，或者通过‘-’将一个授权的缩写与版本名称组合起来。如果必要，更多的自由（参见：http://www.fsf.org/licenses/license-list.html）或开放软件（参见：http://www.opensource.org/licenses/bsd-license.php)）的授权也可以加进来。
- “file LICENSE”或”file LICENCE”中的一个字符串来指名一个在包的最高目录（源或安装目录）中名为“LICENSE”或“LICENCE”的文件。
- 字符串“Unlimited”，用来表示除了相关法律规定外，没有任何散布或使用上的限制

下面是一些标准的授权说明文字：

License: GPL-2
License: GPL (>= 2) | BSD
License: LGPL (>= 2.0, < 3) | Mozilla Public License
License: GPL-2 | file LICENCE

请特别注意，”Public domain”不是一个有效的授权。这个在你包含这些信息时是很重要的！否则，在其它人散布包的拷贝的时候，它甚至不能被合法地纠正。

字段“Description”应当综述包的功能。可以用好几个句子，但只能有一个段落。

字 段“Title”应当对包进行一个简短的描述。一些包的列表有总的大小限制，可能会把标题（Title）截断为 65 个字符。不要使用标记 （markup），不要续行，不要以句号结尾。老版本的 R 使用一个单独的文件“TITLE”来标题信息，现在不允许这么做，并且“Title”字段在文件 “DESCRIPTION”中是必需的。

字段“Author”描述包的作者。它是一段供人阅读的清晰的文字，并不会被用作自动处理（比如读取所有列出的捐献者的 email 地址）。

字段“Maintainer”应当给出一个名字和一个包括在尖括号的有效的 email 地址（用来发送 bug 报告等）。不应当以句号或逗号结尾。

可选字段“Data”用来指明当前版本的包的发行日期。强烈建议使用 ISO 标准的日期格式：yyyy-mm-dd。

可 选字段“Depends”用来指明这个包所依赖的其它包，用逗号隔开。包的名称后面可以加一段用圆括号括起来的注释。注释中应当包含一个比较运算符（在 R 2.7.0 以前的版本中只支持“>=”和“<=”）、空格和有效的版本号（甚至在它们只是包的一部分的时候，也要把包的名字列出来）。如果 你的包是依赖于特殊的版本的时候，你也可以使用特殊的包的名称“R”。比如，如果这个包只能在 R 的 2.4.0 及其以后的版本上运行，就在

“Depends”字段中包含“R(>=2.4.0)”。library 命令和 R 的包的验证程序将使用这个字段，因此不恰当的句法或滥用 “Depends”字段来注释可能会用到的其它程序是不对的。所依赖的其它的环境（R 系统的外部）应当列在“SystemRequirements”字段 或者在一个单独的”README”文件中。在 R 的 INSTALL 程序将检查 R 的版本是否适用于正在安装的包，并且包的列表在加载当前包之前（在版本检查 后）会被加载来，在调用 library 和保存包的代码的镜像（image）或者修复一个超时的加载（lazy-loading）时都会执行这样的操作。
从 R2.7.0 开始，一个包（或“R”）可以多次出现在“Depends”中，但只有第一个出现的才能被早期版本的 R 使用。（不幸的是所有出的都将被检查，所以只有”>=”和”<=”可以使用）

可 选字段“Imports”列出了需要从中引入命名空间但又不需要加载的包的名称，通过”::”或”:::”操作符来访问的命名空间必须列在这或列在 “Suggests”、“Enhances”中（见下面）。最理想的是这个字段包含所有标准的包，并且它的引用包括用作 S4 的包（因为它们的类的定义是可 以改变的，这样，当其发生时，“DESCRIPTION”文件就可以用来决定哪些包需要重新安装）。

可选字段“Suggests”,使用与 “Depends”相同的句法，它列出那些不是必需的包。这些包包括那些仅仅在例子、简述（见 1.4 节）和在函数体中被加载的包。比如，假设包 foo 中有 个例子需要用到包 bar 中的一个数据集，这样在普通的使用中包 bar 就不是必需的，除非想要执行这个例子：最好是加载包 bar，但它不是必需的。
最后，可选字段“Enhances”列出了即将被这个包增强的包，比如为来自这些包的类提供方法。
一般的原则是

- 那些只需要用 library(pkgname)加载命名空间的包必须列到“Imports”字段。
- 那些用 library(pkgname)命令要成功加载一个包而必须一起附带加载的包，必须列到“Depends”字段。
- 那些要成功运行 R 命令 chek 而必需的所有包必须列在“Depends”、“Suggests”、“Imports”中的一个字段中。

特别是，为了尽可能地精减安装，那些只用来为例子或包的说明提供数据的比较大的包应当列在“Suggests”而不是“Depends”。

可选字段“URL”给出了一个由逗号或空格隔开的 URL 列表，比如作者的主页或者有关该软件的其它材料的页面。这些 URL 将在 CRAN 上转换活动的超级链接。
基础和推荐包（即那些在默认和 R 一起安装的或在 CRAN 上推荐安装到 R 的二进制版本中的包）分别有一个值为“base”和“recommended”的“Priority”字段。这种优先权不必使用到“其它”包中。

可 选字段“Collate”（或因操作系统而不同的各种“Collate.OStype”，比如“Collate.windows”）可以用来控制整理一个 包中 R 代码文件的顺序，当它们从原代码安装的时候会连结成一个单一的文件。默认是根据‘C’的顺序来整理。如果出现了该字段，必须在其中列出包中所有的 R 代码文件（要考虑到所有与操作系统相关的子目录，参见 1.1.3），它是一个用空格分开的相对于 R 的子目录的文件路径列表。路径如果包含空格或者引号需要 用括起来。一个恰当的指定了操作系统的 collation 字段(“Collate.unix”或”Collate.windows”)将用来替代 “Collate”。

可选字段“LazyLoad”和“LazyData”控制 R 的对象和数据集（各自的）是否延迟加载（lazy-loading）：如果这个字段的值为“yes”或“true”就会延迟加载，如果设为“no”或“false”就不会延迟加载。（大写字母也是可以的。）
如果你编写的包使用了 methods 包，指定“LazyLoad:yes”。
可选字段“ZipData”控制是否允许 Windows 的内置程序压缩数据目录：如果你的包不使用压缩了的数据目录，将其设置为“no”。
如 果“DESCRIPTION”文件不是完全由 ASCII 构成，就应当包含一个“Encoding”字段指定使用的编码。这个通常是作为 “DESCRIPTION”这个文件本身以及“R”和“NAMESPACE”文件的编码，并且作为“.Rd”文件的默认编码。这时是假设当运行 R 命令 check 时也用该编码，因为从 R 2.8.0 开始，它将作为 CITATION 文件的编码。只有 latin1, latin2 和 UTF-8 这些编码文件是被认为可移植的。（除非的确需要，否则不要指定编码，：这样做会降低包的可移植性。）

可选的字段“OS_type”指定包所运行的操作系统。如果使用该字段，应当是 unix 或 windows，并且表示这个包只能安装在“.Platform$OS.type”指定的平台上。
可选字段“Type”，指定包的类型：参见 1.11 节，包的类型。
注：应当不包括“Built”或“Packaged”字段，因为它们会由包的管理工具自动加载。

1.1.2 “INDEX”文件

可选文件“INDEX”包含一列包中十分有用的对象，给出了他们的名称和描述（像 print 之类通常不显式调用的 方法可能不会包括到里面）。通常该文件是没有的，相应的信息在从源安装和创建包（参见 1.3 节，检查和创建包）时自动地从文件中生成的（使用 tools 包 中的 Rdindex()）。
与其编辑这个文件，还不如将针对包的具体信息放到一个关于包的综述性的手册中（参见 2.1.4，文档打包）或包的简介中（参见 1.4，撰写包的简介）。

1.1.3 包的子目录

子 目录“R”只包含 R 的代码。所安装的代码文件应当以一个 ASCII（小写或大写）字母或数字开始，并且扩展名为“.R”，“.S”，“.q”，“.r”或 “.s”。我们推荐使用“.R”，因为这个扩展名看起来还没有被其它软件使用。有可能用 source()来读入文件中的代码，所以要建立 R 的对象必须用 assignments。注意要将文件名与由该文件创建的 R 的对象联系起来。最理想的是，R 的代码文件应当只指定 R 的对象，并且一定不要调用像 require 和

options 之类的有会带来额外影响的函数。在使用延迟加载的时候，如果要求创建某个对象，这个对象在载入某个包前（参见字段 ”Collate”）就要提前使用该包中的代码，那么在”Depends”字段中的那些包提供的用来创建该对象的函数应提供这样的机制：让那些被创建的对 象不能依赖于这些包，除非通过命名空间来引入它们。（没有使用命名空间的包将会有较少的限制。）

这儿允许有两个例外：如果在 R 的子目录下有一个文 件“sysdata.rda”（保存的 R 的对象的镜象），它将被延迟加载入命名空间/包的环境中，这样做是为了使用方便系统使用系统数据而不是为了让用户 通过 data 来访问它。同时，以“.in”结尾的文件可以放到 R 的目录中，这样就可以通过它使用“configue”脚本一生成适合的文件。
代码 文件中应只使用 ASCII 字符（和控制字符 tab,换页符，换行符，回车符）。其它字符在可以在注释中使用，但些字符在 UTF-8 编码下是不可读的。在安 装时有非 ASCII 字符的对象名，就会出错。在引号括起来的字符串中，任何字节都是可以合用的（但 \uxxxx 是不应被使用的）。但在一些地区，非 ASCII 字符串可能不能使用或显示不正常。包中的各种函数可以被初始化和清除。对于那些没有命名空间的包，比如.First.lab 和.Last.lib(关于有命名空间的包，请参见 1.6.3　加载链)，通常是在名为“zzz.R”的文件中定义。如果.First.lib 在一个包中 被定义，在包被加载和执行 attach 后，它们由参数 libname 和 pkgname 调用。（如果一个带着版本信息的包被安装，那么这个包的名字就包括版 本信息。比如“ash_1.0.9”）。被常用来在.First.lib 中调用 library.dynam 来读取汇编的代码：另一个用途是调用会带来额外 影响的函数。如果在一个包中存在.Last.lib，那么这个包在执行 detach 前，它们会被调用（用一个包的全路径的参数）。没有.Last.lib 函数的包是很少见的：它的一个用处就是调用 library.dynam.unload 来释放汇编代码。

子目录“man”应当只包含包中对象的 Rd(R 文档)格式的文档文件。文件名必须以 ASCII（大写或小写）字母或数字开头，扩展名为“.Rd”（默认）或“.rd”。另外，文件名必须在 URLs：“file://”中是有效的，这就意味着它们必需全部是 ASCII 码，并且不包含“%”。更多信息，参见第二章［编写 R 的文档］。注意，包中 所有用户级的对象都应被编进去。如果一个包 pkg 包含用户级别的对象，但仅在内部（“internal”）使用，那么在包中应当提供一个“pkg- internal.Rd”文件，它们编入所有的这些对象，并且明确说明它们不会被用户调用。R 的标准版本中的 grid 包可以作为例子供参考。注意那些大量 使用内部对象的包，当这些对象没有被编录的时候，应当将这些对象隐藏在一个命名空间中（参见 1.6　包的命名空间）。

子目录“R”和“man”可以包含与特定操作系统相关的子目录，名为“unix”或“windows”。
*****
目录“scr”中加入 的可选文件“Makevars”或“Makefile”是编译后代码的源代码和头文件，当用 R 命令 INSTALL 安装一个包时，Make 用来控制编译并且 连接到加载到 R 中的共享对象。对于 C，C＋＋，FORTRAN77，Fortran 9x,Objective C 和 Objective C++(它们的扩展名分别为“.c”，“.cc”或“.cpp”或“.C”,“.f”,“.f90”或“.f95”,“.m”和“.mm”或“.M””) 这个过程有

默认的变量的规则（当 R 的配置被记录到“R_HOME/etcR_ARCH/Makeconf”中时这些规则就确定了）。对于 C＋＋或 Fortran9x 的文件，我们推荐使用“.h”作为头文件。默认的规则可以通过在文件“src/Makevars”中设置宏来改变（见 1.2　使用 Makevars）。注意这个机制应当通用化，使“src/Makefile”的不再需要***。如果分配了这样一个文件，要非常小心的将它通用化，让它 能够在所有的 R 的平台上使用。它应当有一个恰当的最初目标（一般称为“all”）和（可能是空的）目标“clean”，目标“clean”将清除所有的由 Make(执行 R 安装命令”clean”和 R 安装命令”preclean”)产生的文件。在 Windows 上专用的文件 “src/Makevars.win”的优先级高于“src/Makevars”，并且必须使用文件“src/Makefile.win”。***

子目录“data”是用来存放由包产生的额外数据文件，这些数据文件可以用 data()来加载。目前，数据文件可以有三种类型，其类型由扩展名决定：一般的 R 代码（“.R”或“.r”），表（“.tab”，“.txt”或“.csv”，文件格式请参考?data）或 save()保存的镜象（“.RData” 或“.rda”）。R 的所有端口都使用同样的二进制（XDR）格式，并且可以读取压缩镜象。默认由 save(,compress=TRUE)保存的镜像， 以节省空间。注意，为了在没有加载包的情况下使用数据文件，R 代码应当在数据文件中是封闭的，不要使用包中另外提供的的函数。再也不必在“data”目录 中提供“00Index”文件了，相应的信息会在安装包或者创建包（参见 1.3　检查并创建包）时由文档自动生成。如果你的数据文件比较在，你可能用文件 在”data”子目录中使用一个“datalist”文件来加快安装过程。在该文件中，每个 data()可以找到的主题占一行。如果格式为“foo”,命 令 data(foo)将加载数据“foo”，就用格式“foo”,如果格式为“foo:bar bah”，命令”data(foo)”将加载数据“bar”和“bah”。　***
子目录“demo”是用来保存 R 的脚本文件的（通过 demo()运行），它们用来演示包的一些功能。演示可以是交互的，并且不会被自动检查，所以，如果想要测试演示代码，请使用“tests”子目录下的代 码。脚本文件必须以字母（大写或小写）开头，扩展名为“.R”或“.r”。如果提供了演示，那么“demo”子目录应当提供一个文件“00Index”， 每个演示分别写成一行，给出演示的名称及其描述，名称和描述用空格格开。（这个索引文件是不可能自动生成的。）

子目录“inst”下的内容将被累 加式地复制到包的安装目录下。“inst”的子目录将不会影响那些由 R 使用的目录（目前为 “R”，“data”，“demo”，“exec”，“libs”，“man”，“help”，“html”，“latex”，“R- ex”，“chtml”和“Meta”）。在创建“src”后就拷贝“inst”,这样，“Makefile”就可以创建要安装的文件。注意除 “INDEX”，“LICENSE”/“LICENCE”，“COPYING”和“NEWS”（从 R　2.7.0 开始），包的高级信息文件将不会被安装， 这样用户就不会知道 Windows 和 MacOS X 的被编译了的包的信息（并且用户如果用 R 命令 INSTALL 或者从工具栏执行 install.packages 也看不见）。所以，你如果想要最终用户看到其它信息文件，就应当把它们放到“inst”中。

子目录“tests”是用来存放附加的具体包的测试代码，类似于 R 中发行版本中的具体的测试。测试代码既可以直接由一个“.R”文件来提供，也过以由 “.Rin”文件中的代码来创建相应的”.R”文件（例如：通过收集包中所有的函数对象，然后用临时参数来调用它

们）。运行“.R”代码的结果被写入了文 件“.Rout”中。如果已经存在一个相应的“.Rout.save”文件，就将比较这两个文件，将报告它们的不同之处，并不会引起错误。目录 “tests”将被拷贝到检查区域，并且测试将在这个拷贝上运行，并且将拷贝作为工作目录，还设置了参数 R_LIBS 以确保安装的包在测试期间能够通过 library(pkg_name) 加载。

子目录“exec”可以包含附加的包需要的可执行文件，典型的文件是 shell, Perl 或 Tcl 这些解释程序的脚本。这种机制现在只用在很少的一些包中，并且依然是试验性的。
子目录“po”是用来存放那些与本地化（localization）相关的文件的。参见 1.9　国际化。

1.1.4 包的封装（Package bundles）

有时将几个小的包封装成一个大的包一起发行是比较方便的。（VR 就是一个例子，它包含 4 个包。）在类 Unix 操作系统和 Windowx 上的安装程序都能处理这样的大包。
大包的“DESCRIPTION”文件中有一个“Bundle”字段，没有“Package”字段，如下所示：

```
Bundle: VR
Priority: recommended
Contains: MASS class nnet spatial
Version: 7.2-36
Date: 2007-08-29
Depends: R (>= 2.4.0), grDevices, graphics, stats, utils
Suggests: lattice, nlme, survival
Author: S original by Venables & Ripley.
R port by Brian Ripley <ripley@stats.ox.ac.uk>, following earlier
work by Kurt Hornik and Albrecht Gebhardt.
Maintainer: Brian Ripley <ripley@stats.ox.ac.uk>
BundleDescription: Functions and datasets to support Venables and
Ripley, 'Modern Applied Statistics with S' (4th edition).
License: GPL-2 | GPL-3
URL: http://www.stats.ox.ac.uk/pub/MASS4/
```

“Contains”字段列出了大包中包含的小包（空格分开），它们将按照它们的名字分别放到不同的子目录中。在创建和安装期间，包（后面的包指的是小包）将按指定的顺序安装。请确认列表的顺序与它们之间的依赖关系是对应的。
除了“DESCRIPTION”文件被替换为“DESCRIPTION.in”外，大包中的小包都是标准的包。在“DESCRIPTION.in”中只包含了附加到大包中文件“DESCRIPTION”的字段。比如：

```
Package: spatial
Description: Functions for kriging and point pattern analysis.
Title: Functions for Kriging and Point Pattern Analysis
```

除了“DESCRIPTION”文件和命了名的包之外，在大包中的所有文件，都将被忽略。
大包中的“DESCRIPTION”文件中的“Depends”字段将列出所有小包的从属（类似于“Imports”和“Suggests”），并且文件“DESCRIPTION.in”就不应再包含这些字段了。

1.2 配置和清理

注意节的大部分都是针对 Unix 的：参考后面关于 R 的 Windows 端口的评价。在安装包之前，如果你的包需要针对特定的操作 系统作一些配置，你可以在包中安排一个可执行（Bourne shell）脚本文件“configure”，在运行其它程序前，R 命令 INSTALL 将首先执行（如果有）该文件。该脚本可以由 Autoconf 生成。 也可以由你自己写。这样就可以用来检测是否提供了非标准的程序（library），让错误的代码在安装的时候就被发现，而不是在包被编译后或者使用时才给 出错误提示。总之，你可以在你的扩展包中最大限度地发挥 Autoconf 的作用（包括变量代换，搜索程序库等）。

在 Unix 及其类似的操作系统 上，如果在运行 R 命令 INSTALL 时给出了选项“--clean”，那么在程序的最后一步将运行脚本（Bourne shell）程序”cleanup”。在用 R 命令 build 准备从源创建包时也是这样。可以用它来清除包的源目录树，特别是用来删除所有由“configure”创建的文件。

比如，假设我们想使用一个（C 或 FORTRAN）程序库 foo 的某些功能。我们可以用 Autoconf 来创建 一个配置脚本，用来查找程序库，将如果找到了，就将变量 HAVE_FOO 设置为 TRUE，如果没找到，就设为 FALSE，然后将这个值写入到输出文件（用 HAVE_FOO 的值来替换输入文件中“@HAVE_FOO@”的实例）。例如程序库 foo 可以链接到函数 bar（即使用”-lfoo”），可以在 “configure.ac”中使用（假设是 Autoconf 2.50 或更新的）

```
AC_CHECK_LIB(foo, fun, [HAVE_FOO=TRUE], [HAVE_FOO=FALSE])
AC_SUBST(HAVE_FOO)
......
AC_CONFIG_FILES([foo.R])
AC_OUTPUT
```

相应的 R 的函数“foo.R.in”可以定义为：

```
foo <- function(x) {
if(!@HAVE_FOO@)
stop("Sorry, library ’foo’ is not available"))
...
```

如果程序库 foo（和想要的功能）没找到，从该配置文件创建的 R 源文件“foo.R”类似

```
foo <- function(x) {
if(!FALSE)
stop("Sorry, library ’foo’ is not available"))
...
```

上面的代码有效地将该函数变为不可用。

对于那些可用或者不可用功能，可以分别使用不同的程序块。

在编译 R 或者你的包的时候，你很可能需要确保样同的 C 编译器和编译标识可以用在“configure”测试中。在 Unix 下，为了实现这个目的，你可以在“configure.ac”文件的前面部分写入下列程序段：

```
: ${R_HOME=‘R RHOME‘}
if test -z "${R_HOME}"; then
echo "could not determine R_HOME"
exit 1
fi
CC=‘"${R_HOME}/bin/R" CMD config CC‘
CFLAGS=‘"${R_HOME}/bin/R" CMD config CFLAGS‘
CPPFLAGS=‘"${R_HOME}/bin/R" CMD config CPPFLAGS‘
```

（当将脚本文件作为 R 命令 INSTALL 一部分的时候，为了能够使用正确的 R 的版本，有必要使用“${R_HOME}/bin/R”，而不要使用“R”）
注 意本文档的早期版本推荐直接从“R_HOME/etcR_ARCH/Makeconf”获得（使用 grep 或 sed）配置信息，这样就只能测试那些作为字 面值记录在那的变量。你可以使用 R 命令 config 来获取基本配置变量的值或头文件和程序库标识，它们都是联结到 R 所必需的。更详细信息参考 R 命令 config—help。（从 R 2.6.0 开始，也可以在 Windows 上运行。）

注意由 R 决定的 FLIBS 必须用来确保 FORTRAN77 代 码能够在所有 R 平台上运行。不必调用（并且由此，例如，从 ACX_BLAS 移除）Autoconf 的宏 AC_F77_LIBRARY_LDFLAGS，它 将重写 FLIBS。（目前版本的 Autoconf 实际上允许一个已经存在的 FLIBS 来改写 FORTRAN 连结标识的测试。同时，目前版本的 R 可以检测外 部的 BLAS 和 LAPACK 程序库。）

要检查外部的 BLAS 程序库，可以使用官方的 Autoconf 宏包中的 ACX_BLAS 宏，如下所示：

```
F77=‘"${R_HOME}/bin/R" CMD config F77‘
AC_PROG_F77
FLIBS=‘"${R_HOME}/bin/R" CMD config FLIBS‘
ACX_BLAS([], AC_MSG_ERROR([could not find your BLAS library], 1))
```

注 意像由 R 决定的 FLIBS 必须用来确保 FORTRAN77 代码能够在所有 R 平台上运行。不必调用（并且由此，例如，从 ACX_BLAS 移 除）Autoconf 的宏 AC_F77_LIBRARY_LDFLAGS，它将重写 FLIBS。（目前版本的 Autoconf 实际上允许一个已经存在的 FLIBS 来改写 FORTRAN 连结标识的测试。同时，目前版本的 R 可以检测外部的 BLAS 和 LAPACK 程序库。）

你应当记住配置脚本文件在 Windows 系统上无法正常运行（尽管简单的 shell 脚本可以运行，但如果是由 Autoconf 生成的脚本一般是不能运行的）。如果你的包是要给公众使用，请给那些使用非 Unix 平台的用户足够的信息，好让他们手动配置，或者提供一个能在 Windows 平台下使用的脚本文件 “configure.win”。（那也可以放置一个可选的 cleanup.win”脚本，它们都应当是可以由 ash 执行的脚本文件，ash 是一个 Bourne-style sh.的最小化版本。）

在一些极少见的情况下，配置和清理脚本文件需要知道包的安装位置。比如，在包中使用了 C 语言代码，并且建立了两个共享对象或 DLL。通常，被 R 动态地加载的对象是与该附加对象相联系的。在一些操作系统中，当 R 动态地调用一个对象时，可以将其所依赖 的其它对象的地址指定给该象。这意味着每个用户不必设置环境变量 LD_LIBRARY_PATH(或其它类似的变量)，而自动加载该附加对象。另一个例子 是当包在安装在运行时使用的文件的时，它们的地址会在安装的时候被写入 R 的数据结构中（如 SJava 包中的 Java Archive 文件）。最高级的程序库目录（由“-l”参数指定的）和包的目录本身可以通过 shell/环境变量 R_LIBRARY_DIR 和 R_PACKAGE_DIR 指定线安装脚本文件。另外，包的名称（如“survival”或“MASS”）也可以从 shell 变量 R_PACKAGE_NAME 获得。

1.2.1 使用“Makevars”