如何写 R 的程序包

综述

* 该指南将介绍如何在Unix/Linux下创建在Unix/Linux下使用的R的包（package）。如果你要将你所创建的Unix/Linux下使用的包发布在CRAN，CRAN将为你创建在windows下使用的包。 创建一个在Windows下使用的包稍微有点复杂，虽然过程差不多，但需要安装Perl等其它软件（需要安装的软件请参考R语言中文社区）。
* 更详细的创建包的内容请参考手册《Writing R Extensions》
* 所有的函数和变量的名称都必须避免与 R 的内部函数名相冲突。另外，在变量名称中的“.”将被当作分隔符，比如“logLikelihood”是允许的，但“log.likehood”就不合法（因为已经有内部函数名称 log）。
* 尽可能避免使用全局变量（global variables），如果一定要使用，不要使用“a<<-b”来赋值，使用：

```
assign("a",b,.GlobalEnv)
a <- get("a", pos=globalenv())
```

一般来说，全局变量最好使用长的有意义的命字，可以避免潜在的命名冲突。如使用“Xinput”而不用“X”。
* 如果将包发布在 CRAN 上，CRAN 就会自动为你的包创建一个完全的 PDF 手册，如果你不发布，在你使用命令”R CMD build …”（见下面）的时候，也会自动为你创建一部分。

创建过程:

1.) 在你的 Unix/Linux 控制台（console）上创建一个文件夹，用来安装你将要创建的包。比如：myrlibrary

```
mkdir mylibrary
```

2.) 创建另一个文件夹，用来创建你的包,并进入该文件夹。比如：myrpackages

```
mkdir myrpackages
cd myrpackages
```

3.) 运行 R，并且在 R 中定义包中要用到的所有函数。
4.) 在 R 中用 package.skeleton()函数为你的包创建的文件夹结构和代码模板。如：

```
package.skeleton(list=c("<function 1>","<function 2>", ... ,"<function n>"),
name="<package.name>")
```

5.) 现在“package.skeleton()”在当前目录创建了一个文件夹<package.name>在其中还创建了包中要用到的子文件夹。进入<package.name>/man，在参考手册中填入相应的信息。要小心使用符号'_' 和 '^'它们可能会被当作 latex 命令。 Keywords: 可以在 R 中用下列命令来查看哪些 keywords 是可用的:

```
file.show(file.path(R.home("doc"), "KEYWORDS"))
```

6.) 填写<package.name>/DESCRIPTION. License 字段你可以填入'GPL (>= 3)'.
7.) 为了避免用户直接访问函数，建立一个名为<package.name>/NAMESPACE 的文件 (没有扩展名 *.txt) 并在文件中加入所有可见的(VISIBLE)函数，如下

```
export(<function1>,<function2>)
```

现在函数 '<function1>'和 '<function2>' 是可见的，可用由用户直接使用。其它的函数只能

由 R 访问，用户不能直接访问。
比如，用户安装该包后执行下列程序：

```
library(<package.name>)
function1        # 可见的
function3        # 不可见的，因为没有包含在 NAMESPACE 中
<package.name>:::function3   # 让 function3 可见
```

8.) 在控制台下创建包：

```
R CMD build <package.name>
```

9.) 在控制台下检查包：

```
R CMD check <package.name>
```

修正所有的错误和警告，只有修正了所有的错误和警告，才能发布在 CRAN 上。
10.) 在你自己的系统上安装包，在控制台中执行:

```
R CMD INSTALL -l <TARGET DIRECTORY> package.tar.gz
```

这里的 <TARGET DIRECTORY> 是你在 1)所建立的文件夹。

在你的 R 代码中包含 R 代码：

* 注意在 C 语言中，指针（pointers）是针对变量来说的。变量“V”就是一个指向内存（RAM）某个地址的指针，“&V”是指该变量的地址，“V”指该变量的实际值。
* 如果使用“void”函数（没有返回值的函数）。我们需要的输出将保存在内存中。其输出变量也可以是函数的参数（在我们的例子中为“out”）
* 因为函数的参数只能使用指针，因此 C 语言的函数的参数的类别只能由星号来区分，即使用“double”、“int”等。
* 如果你在 C 语言的函数中使用了自己的变量，比如”int x”，在代码的最后记得用 free(x) 来清除内存。
* 例如：我们想要在 R 中用一个 C 语言函数加入两个 double number，建立一个文本文件“plus.c”并且按如下编辑：

```
#include<R.h>
#include<Rmath.h>
#include<Rdefines.h>
#include<Rinternals.h>

void plus(double* a, double* b, double* out)
{
    out = *a+*b;
}
```

* 保存并在控制台下编译该文件：

```
R CMD SHLIB plus.c
```

如果代码没有出错，在当前目录下将会生成两个文件“plus.o”，“plus.so”。
* 在 R 中：用“dyn.load”来加载编译好的代码，并使用“.C”来访问该函数，即：

```
dyn.load("plus.so")
x <- 5.234
y <- 87.234
output <- double(1)
```

```
aplusb <- .C("plus",
as.double(x),
as.double(y),
as.double(output))[[3]]
aplusb
```

* 在我们的例子中，前两个参数为输入参数，第三个参数是用来存储输出结果的。因为我们是将第三个参数作为输出，所以就用“[[3]]”来引用它。如果我们使用的函数在另一个包中，可以使用参数“package”来指定该包（参考.C 的帮助）。

* 在上例中，我们用到了变量类型转换，R 和 C 相对应的变量类型如下表所示：

R storage mode	***C type***
logical	int *
integer	int *
double	double *
complex	Rcomplex *
character	char **
raw	unsigned char *

*也可以用“.Call”来代替代“.C”，“.C”允许输入更多的参数。它们的区别参见帮助文档。
*更多的细节可以参考《Writing R Extensions》
http://cran.r-project.org/doc/manuals/R-exts.pdf

*将上述代码加入到 R 的包中：在运行“package.skeleton()”后，在<package.name>目录下，建立了一个“src”子目录，在该目录下，也有“man”和“R”目录，将函数和 C 代码（plus.c）拷贝到该目录，继续执行 5）。

将包上传到 CRAN:

- 上传文件 .tar.gz (根据手册，用'anonymous' 作为登陆名 E-mail 地址作为密码)到
ftp://cran.R-project.org/incoming/
发一封上传文件的提示电子邮件到 cran@r-project.org, 收件人为来自 Vienna 的 Kurt Hornik

欢迎大家加入google上的R语言中文社区

本文由 胡荣兴（hurongxing@126.com）译自：
http://www-m4.ma.tum.de/Diplarb/howtorpackage.html