R 语言编程：基于 tidyverse

第 28 讲　开发 R 包

张敬信

2022 年 3 月 28 日

哈尔滨商业大学

编程中，提倡：

- 将实现一个功能自定义为一个函数，这样就可以方便自己和他人重复使用；
- 将完成一项工作，组织在一个 R 项目方便统一管理，里面包含一系列好用的自定义函数；
- 如果再进一步，想让你的 R 项目变成通用的工作流，可以方便自己和他人解决同类问题，就是将 R 项目变成 **R 包**。

R 包将代码、数据、文档和测试捆绑在一起，便于自己复用且很容易与他人分享，同时也为参与和繁荣 R 社区做出贡献。

Hadley 等开发的 devtools 系列包，可以说让如今的 R 包开发变得非常简单和自动化，其理念就是：让开发包的整个工作流程尽可能地用相应函数自动化实现，让你把时间花在思考你想让你的包做什么，而不是思考包的结构的细枝末节。

R 包有五种形态：**源码、捆绑、二进制、已安装、载入内存**。前三种形态是开发和发布 R 包所涉及的，后两种形态是大家已经熟悉的安装包和加载包。

一. 准备开发环境

安装专为开发 R 包打造的 devtools 包，顺便会安装 usethis 包（自动化设置包和项目）、roxygen2 包（各个函数提供文档）、testthat 包（进行单元测试）等。

Windows 系统从源码构建 R 包所需的工具集，称为 Rtools[1]，需要从 CRAN 下载相应版本并按默认选项安装，然后在 R 中执行：

```
writeLines('PATH="${RTOOLS40_HOME}\\usr\\bin;${PATH}"',
           con = "~/.Renviron")
```

重启 R，检查：

```
devtools::has_devel()   # 或者 Sys.which("make")
```

[1] Mac 系统需要安装 Xcode 命令行工具，并注册为苹果开发者.

二. 编写 R 包工作流

假设你有开发一个新包的想法，首先是为它寻找和挑选一个合适的包名，`available` 包可以为你提供灵感和检查名字是否可用。

我计划为常用的数学建模算法开发一个 R 包：`mathmodels`，让国内做数学建模的高校学生、教师能够不再依赖 Matlab（体积庞大、非开源免费、还禁用部分高校）。

1. 创建 R 包

建议让 R 包包含 Git 版本控制，并且能在远程 Github 仓库同步（相当于发布在 Github）。

如 Git 节所做的一样：先在 Github 建立远程同包名的仓库，再在本地新建带 Git 版本控制的同包名的 R 项目。

接着，从该 R 项目开始创建 R 包：

```
library(devtools)
create_package(getwd())     # 从当前路径创建 R 包
```

选择可重写 `mathmodels.Rproj`, 则在本地创建一个初始的源码 R 包结构:

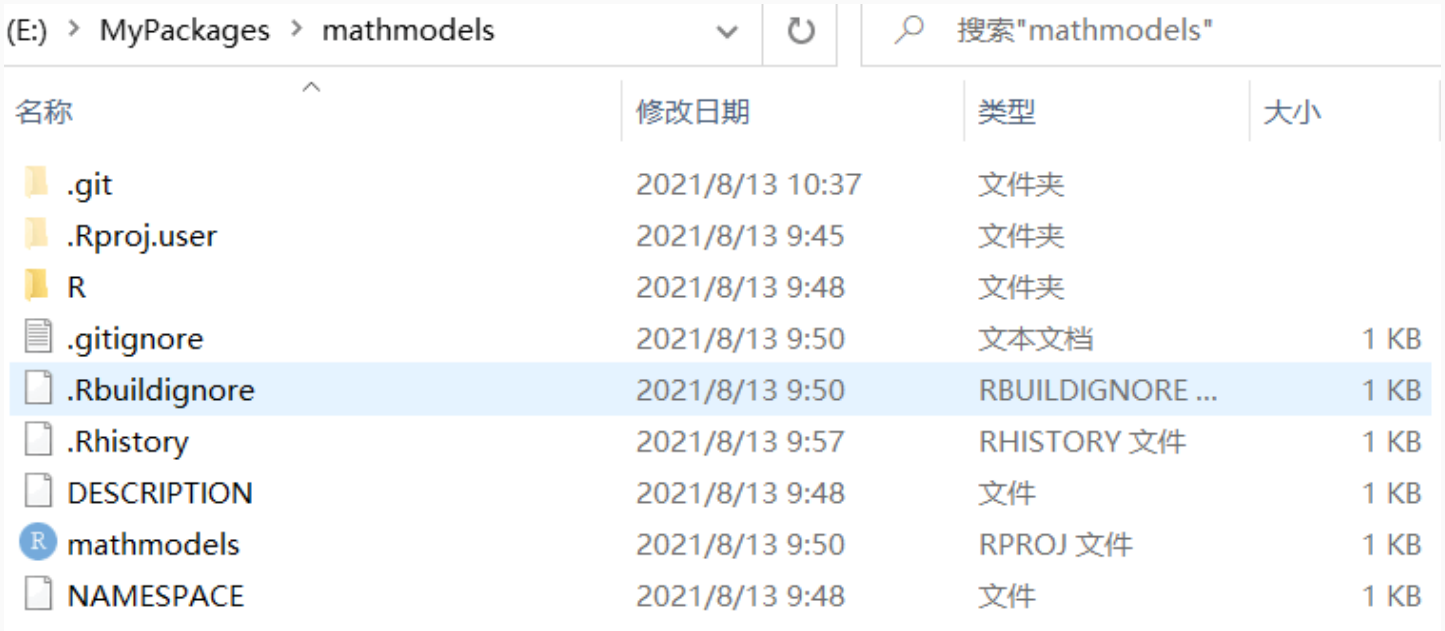


图 1: R 包的源码文件

构成 R 包的文件[2]：

- .gitignore 和.Rbuildignore: 包含在 Git 或 R 包构建时应该忽略的文件
- DESCRIPTION: 关于你的包的元数据
- NAMESPACE: 声明你的包对外输出的函数和从其他包导入的外部函数，将用 document() 自动再生成
- R: 包含所有你自定义的函数
- mathmodels.Rproj: R 项目文件

至此，初步的开发 R 包的框架已经搭建完成，并且已经与远程仓库建立连接，后续任何开发更新，都能很容易提交到 Github 仓库[3]

[2]其中需要编辑改写的文件，都将用 devtools::document() 自动生成.

[3]如 Git 节做法一样: Staged -> Commit -> Push.

2. 添加函数

R 包最核心的部分就是你自定义的函数，其余都是配套的使用说明、保证函数可运行的依赖、数据集等。

R 文件夹包含了你所有的自定义函数，每个函数都保存为一个同名的.R 文件。

现在来写第一个函数：**AHP( ) 实现层次分析法。**

R 包中的自定义函数，本质上与普通的自定义函数并没有不同，只是额外需要注意：

- 增加函数注释信息，将用于生成函数帮助
- 调用其他包中的函数时，用包名前缀，不加载包
- 永远不要使用 `library( )` 或 `require( )`, 永远不要使用 `source( )`

执行下面语句，自动在 R 文件夹创建并打开 AHP.R：

```
use_r("AHP")
```

将调试通过的函数代码放进来：

```r
AHP = function(A) {
  rlt = eigen(A)
  Lmax = Re(rlt$values[1])   # Maximum eigenvalue
  # Weight vector
  W = Re(rlt$vectors[,1]) / sum(Re(rlt$vectors[,1]))
  # Consistency index
  n = nrow(A)
  CI = (Lmax - n) / (n - 1)
  # Consistency ratio
  # Saaty's random Consistency indexes
  RI = c(0,0,0.58,0.90,1.12,1.24,1.32,1.41,1.45,1.49,1.51)
  CR = CI / RI[n]
  list(W = W, CR = CR, Lmax = Lmax, CI = CI)
}
```

光标放在函数体内，点击 `Code -> Insert roxygen skeleton`，自动插入函数注释信息模板。

为本函数编写的注释信息，每行都是以 `#'`开头，`@`引导的关键词：标题、描述、参数、返回值、工作示例分别填写相应内容。

```
#' @title AHP: Analytic Hierarchy Process
#' @description AHP is a multi-criteria decision analysis method developed
#' by Saaty, which can also be used to
#' determine indicator weights.
#' @param A a numeric matrix, i.e. pairwise comparison matrix
#' @return a list object that contains: W (Weight vector), CR (Consistency ratio),
#' Lmax (Maximum eigenvalue), CI (Consistency index)
#' @export
#' @examples
#' A = matrix(c(1,   1/2, 4, 3,   3,
#' 2,   1,   7, 5,   5,
#' 1/4, 1/7, 1, 1/2, 1/3,
#' 1/3, 1/
#' 1/3, 1/
#' AHP(A)
```

```
#' A = matrix(c(1,   1/2, 4, 3,   3,
#' 2,   1,   7, 5,   5,
#' 1/4, 1/7, 1, 1/2, 1/3,
```

有了上述帮助信息，执行文档化：

```
document()
```

将自动生成函数帮助，实际上是调用 roxygen2 包生成 man/AHP.Rd.

该文件在 RStudio Help 窗口显示就如我们平时用? 函数名查看帮助所看到的一样：

AHP: Analytic Hierarchy Process

Description

AHP is a multi-criteria decision analysis method developed by Saaty, which can also be used to determine indicator weights.

Usage

```
AHP(A)
```

Arguments

A	a numeric matrix, i.e. pairwise comparison matrix

Value

a list object that contains: W (Weight vector), CR (Consistency ratio), Lmax (Maximum eigenvalue), CI (Consistency index)

Examples

```
A = matrix(c(1,   1/2, 4, 3,   3,
             2,   1,   7, 5,   5,
             1/4, 1/7, 1, 1/2, 1/3,
             1/3, 1/5, 2, 1,   1,
             1/3, 1/5, 3, 1,   1), byrow = TRUE, nrow = 5)
AHP(A)
```

如果是新包，建议加上 `@export` 以导出你的函数，这样做文档化时会自动将该函数添加到 `NAMESPACE` 文件。

导出的函数也是给安装你的包的用户使用的函数，不导出的函数叫作内部函数，只供包里的其他函数使用。

3. 编辑元数据

每个包都必须有一个 DESCRIPTION 文件，它是用来存放关于你的包的重要元数据[4]。

点开 DESCRIPTION 文件，包名、编码等部分信息是自动生成的，可编辑标题（单行文字）、版本号、作者、描述（一段文字）、网址等信息，导入、许可等信息更建议通过命令添加。

[4]这也是一个 R 包的决定性特征：RStudio 和 devtools 认为任何包含 DESCRIPTION 的目录都是一个包.

```
Package: mathmodels
Title: Implement Common Mathematical Modeling Algorithms with R
Version: 0.0.1
Authors@R:                                  # 多个作者用 c() 合并
    person(given = "Jingxin",
           family = "Zhang",
           role = c("aut", "cre", "cph"),  # 作者, 维护者, 版权人, 还有"ctb" 贡献者
           email = "zhjx_19@hrbcu.edu.cn")
Description: Mathematical modeling algorithms are classified as evaluation,
  optimization, prediction, dynamics, graph theory, statistics,
  intelligence, etc. This package is dedicated to implementing various
  common mathematical modeling algorithms with R.
License: AGPL (>= 3)
URL: https://github.com/zhjx19/mathmodels
BugReports: https://github.com/zhjx19/mathmodels/issues
Encoding: UTF-8
LazyData: true
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.1.1
Imports:
    deSolve
```

(1) 版本号

通常是三位：大版本．小版本．补丁版本，按数值大小递进，开发版本一般从 9000 开始：0.0.1.9000。

(2) 依赖包

Imports: 下所列的包是必须存在，你的包才能工作，别人安装你的包时，也会自动安装这些包；

Suggests: 下所列的包是建议包[5]，不会随你的包自动安装，所以在使用之前通常需要检查是否存在：

```
if (requireNamespace("pkg", quietly = TRUE)) {
    pkg::f()
}
```

[5]比如案例数据集、运行测试、用于 Vignette 等.

建议用命令方式添加依赖包或建议包：

```
use_package("deSolve")  # 还有参数 min_version 指定最低版本
use_package("deSolve", "Suggests")
```

还有 `@importFrom dplyr "%>%"` 从某包导入单个函数或符号；

`Depends:` 要求最低 R 版本。

(3) **选择许可**

这里用命令方式选择比较流行的 GPL-3 [6]开源许可：

```
use_agpl3_license()
```

- `LazyData: true` 确保加载包时自动惰性加载（使用时才载入内存）内部数据集。

[6]GPL 许可规定任何将你的代码以捆绑形式发布的人必须以与 GPL 兼容的方式对整个捆绑进行许可. 此外，任何分发你的代码的修改版本（衍生作品）的人也必须提供源代码. GPL-3 比 GPL-2 更严格一些，关闭了一些旧的漏洞.

4. 使用数据集

包中包含数据集有三种主要方式，这取决于你想用它做什么以及谁能够使用它。

(1) 外部使用

如果你想存储二进制数据并使其对用户可用，就把它以.rda 格式放在data/中，适合放示例数据集。

先把数据集读入到当前变量，比如企鹅数据集 penguins，再执行：

```
use_data(penguins)        # 参数 compress 可设置压缩格式
```

内部数据集就像你的函数一样需要做文档化：文档化数据集的名称并将其保存在 R/中。

先创建：

```
use_r("penguins")
```

再编辑该数据集的注释信息，将用于生成该数据集的帮助：

```
#' @title Size measurements for adult foraging penguins near Palmer Station, Antarctica
#' @description Includes measurements for penguin species, island in Palmer Archipelago,
#' size (flipper length, body mass, bill dimensions), and sex.
#' @docType data
#' @usage data(penguins)
#' @format A tibble with 344 rows and 8 variables:
#' \describe{
#'   \item{species}{a factor denoting penguin species}
#'   \item{island}{a factor denoting island in Palmer Archipelago, Antarctica}
#'   \item{bill_length_mm}{a number denoting bill length (millimeters)}
#'   \item{bill_depth_mm}{a number denoting bill depth (millimeters)}
#'   \item{flipper_length_mm}{an integer denoting flipper length (millimeters)}
#'   \item{body_mass_g}{an integer denoting body mass (grams)}
#'   \item{sex}{a factor denoting penguin sex (female, male)}
#'   \item{year}{an integer denoting the study year (2007, 2008, or 2009)}
#' }
#' @references This dataset referenced from the palmerpenguins package.
```

在关键词引导下，编辑数据集标题、描述、变量说明、来源、示例等信息。还有 `@source` 为你自己获得数据的来源，通常是一个 `url{}`.

注意，永远不要 `@export` 一个数据集。

有了上述帮助信息，执行文档化：

```
document()
```

(2) 内部使用

如果你想存储处理过的数据，但不向用户提供，就把它以.rda 格式放在R/中，适合放你的函数需要的数据。

同样的操作，除了设置 internal 参数为 TRUE:

```
use_data(penguins, internal = TRUE)
```

(3) 原始数据

如果你想展示加载/处理原始数据的例子，就把原始数据文件放在 inst/extdata 中，安装包时，inst/中的所有文件（和文件夹）都会被上移一级目录（去掉 inst/）。

要引用 inst/extdata 中的数据文件（无论是否安装）：

```
system.file("extdata", "mtcars.csv", package = "readr",
            mustWork = TRUE)
```

参数 mustWork = TRUE 保证若文件不存在，不是返回空字符串而是报错。

另外，通常你的数据集是你搜集的原始数据经过处理的版本，Hadley 建议额外将原始数据和处理过程代码放入 data-raw/，这只是便于你将来更新或重现数据，捆绑 R 包时它们是不需要的，所以需要添加到.Rbuildignore，use_data_raw() 能帮你自动完成。

5. 单元测试

测试是开发 R 包的重要部分，可以确保你的代码更稳健，能成功地做你/用户想做的事情。

测试的一般原则是，设想你的函数各种可能遇到的情况，是否都能得到预期的结果。策略之一是每当你遇到一个 bug，就为它写一个测试，以检查将来是否会出现这种情况。

虽然执行 `load_all()` 模拟加载你的包，可以在控制台做一些函数测试，但更好的做法是采用正式的自动化测试：`testthat` 包提供的单元测试。

先初始化包的单元测试：

```
use_testthat()
```

它将 `Suggests: testthat` 添加到 `DESCRIPTION`，创建目录 `tests/testthat/`，并添加脚本 `tests/testthat.R`。然而，真正的测试还是要靠你自己来写！

先打开或创建针对某函数的测试文件：

```
use_test("AHP")
```

测试文件是由若干个 test_that() 构成，第一个参数是测试的描述，测试内容是大括号括起来的代码块，一般是比较函数返回值与期望值，是否（近似）相等，是否符合类型等，比如

```
test_that("AHP weights and type", {
  A = matrix(c(1,   1/2,
               2,   1), byrow = TRUE, nrow = 2)
  rlt = AHP(A)
  expect_equal(rlt$W, c(0.3333, 0.6667), tolerance = 0.001)
  expect_type(rlt, "list")
})
```

然后，执行测试（全为 PASS 表示通过测试）：

```
test()
```

单元测试没问题，再执行 R CMD check 检测：

```
check()
```

该命令可能需要一些时间，并在控制台中产生一个输出，关于潜在错误、警告、注意的具体反馈，我们希望三者都是 0.

通过检测的 R 源码包已经可以在自己电脑安装使用了：

```
install()                          # 安装包
library(mathmodels)
# some code
```

按照标准的步骤：Staged -> Commit -> Push 推送到 Github 远程仓库，就是成功发布到 Github，别人也已经可以从 Github 安装和使用你的 R 包。

三. 发布到 CRAN

如果你想让你的包在 R 社区分享，则需要把它提交到 CRAN，这比发布在 Github 上要做多得多的工作：

- 选择一个三位版本号：大版本. 小版本. 补丁
- 检测是否符合 CRAN 政策，在至少两种操作系统执行 R CMD check，并准备 cran-comments.md 文件加以说明
- 编写 README.md 和 NEWS.md
- 用 devtools::build() 从源码包创建捆绑包 tar.gz 格式
- 向 CRAN 提交包
- 通过更新版本号为下一个版本做准备
- 发布新的版本

但这是值得的，因为只有发布到 CRAN，广大 R 用户才能更容易发现和使用你的 R 包。

CRAN 政策，除了基本的规范流程要求之外，其他主要注意事项：

- 你的包的维护者的 Email（长期）可用，CRAN 要确保能联系到你
- 必须在 DESCRIPTION 中明确指出版权人，若包含外部源代码必须兼容许可
- 要求你的包在至少两个操作系统平台上通过 R CMD check，建议也在 R 开发版本上通过 R CMD check
- 禁止做替用户做外部修改，不要写到文件系统、改变选项、安装包、退出 R、通过互联网发送信息、打开外部软件等
- 不要过于频繁地提交更新，建议最多每 1-2 个月提交一次新版本

1. CRAN 检测

在多个操作系统做 R CMD check 都要保证错误、警告、注意为 0，但新包第一次提交必有一个注意，提醒 CRAN 这是一个新的提交。这无法消除，可在 cran-comments.md 中注明这是你第一次提交。

rhub 包可以帮助你在多个操作系统做 R CMD check，还能自动生成检测结果的描述，并用于生成 cran-comments.md。

第一次使用 rhub, 需要先验证你的 Email 地址:

```
library(rhub)
validate_email("zhjx_19@hrbcu.edu.cn")
```

这将向你的该邮箱发送一个 token 码，在提示中输入将绑定你的 Email.

在多个操作系统上对你的 R 包执行 R CMD check，只需运行：

```
results = check_for_cran()
```

检测过程会有一点点长，你的 Email 会陆续收到 3 个邮件，其中的链接详细反馈了测试在三个不同操作系统上的表现。将检测结果赋值，是方便查看检测的概述结果：

```
results$cran_summary()
```

```
For a CRAN submission we recommend that you fix all NOTEs, WARNINGs and ERRORs.
## Test environments
- R-hub windows-x86_64-devel (r-devel)
- R-hub ubuntu-gcc-release (r-release)
- R-hub fedora-clang-devel (r-devel)

## R CMD check results
> On windows-x86_64-devel (r-devel), ubuntu-gcc-release (r-release), fedora-clang
-devel (r-devel)
  checking CRAN incoming feasibility ... NOTE
  Maintainer: 'Jingxin Zhang <zhjx_19@hrbcu.edu.cn>'

  New submission

0 errors √ | 0 warnings √ | 1 note x
> use_cran_comments()
√ Setting active project to 'E:/MyPackages/mathmodels'
√ Writing 'cran-comments.md'
√ Adding '^cran-comments\\.md$' to '.Rbuildignore'
* Modify 'cran-comments.md'
```

再生成 cran-comments.md, 稍加修改就能使用:

```
use_cran_comments()          # usethis 包
```

2. 编写 README、NEWS

若你的包发布在 GitHub，则有必要编写 README.md。它是你的包的主页和欢迎页，包含如何简单使用。执行：

```
use_readme_rmd()
```

则生成并打开 README.Rmd 模板，编辑相应内容即可。

NEWS 文件用在每次更新包的版本时，用来描述了自上一版本以来的变化，执行：

```
use_news_md()
```

则自动生成并打开 NEWS.md，按 markdown 语法无序列表语法编辑内容即可。

3. 捆绑包与提交

源码包需要 Build 为捆绑包 (tar.gz)，才能往 CRAN 提交，执行：

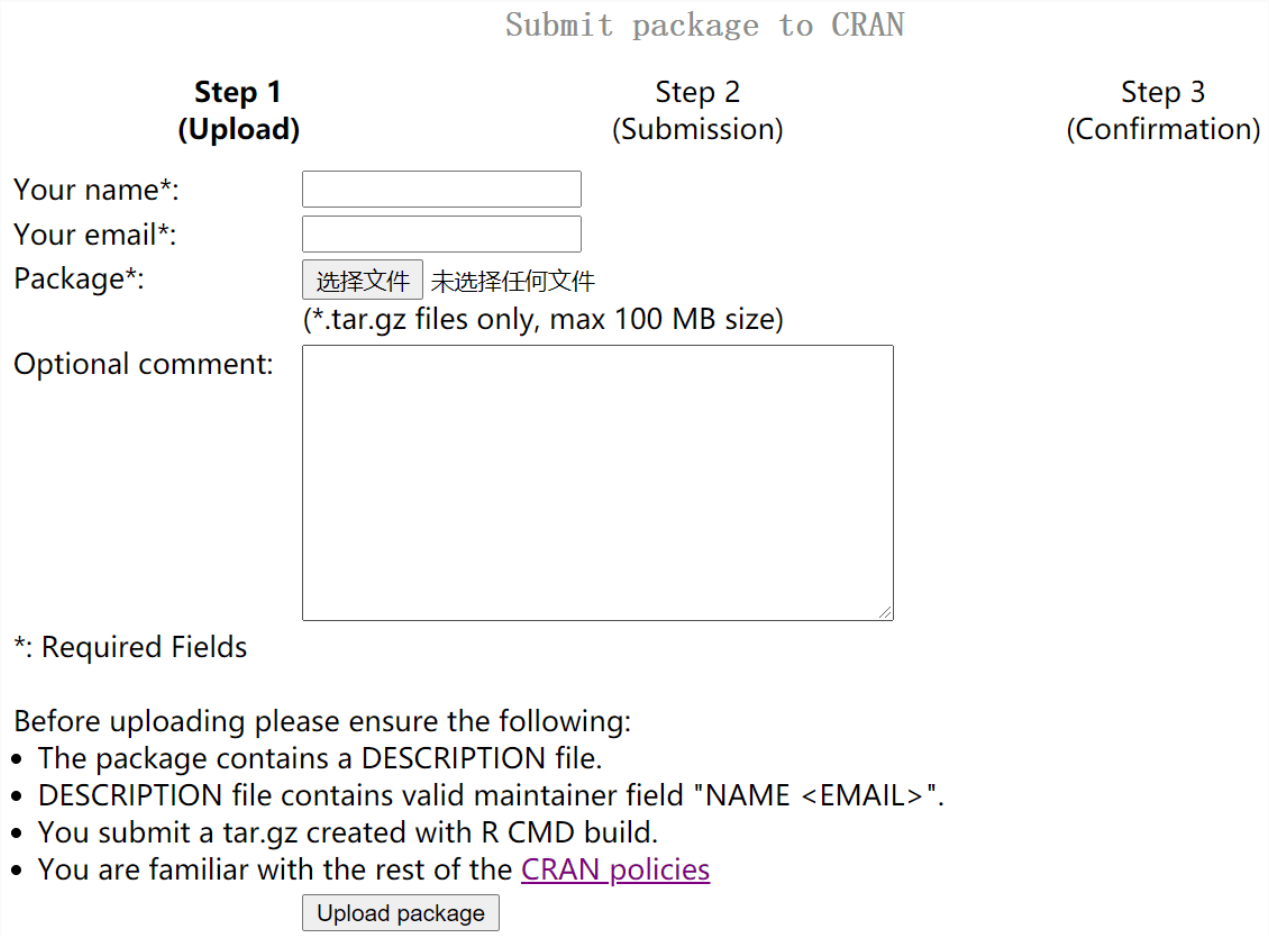
```
build()
```

```
√  checking for file 'E:\MyPackages\mathmodels/DESCRIPTION' ...
-  preparing 'mathmodels':
√  checking DESCRIPTION meta-information ...
-  installing the package to build vignettes
√  creating vignettes (2.6s)
-  checking for LF line-endings in source and make files and shell scripts
-  checking for empty or unneeded directories
-  building 'mathmodels_0.0.1.tar.gz'

[1] "E:/MyPackages/mathmodels_0.0.1.tar.gz"
```

有了 mathmodels_0.0.1.tar.gz 和 cran-comments.md，终于可以向 CRAN 提交了。

打开https://cran.r-project.org/submit.html, 按要求提交即可：



提交后，你会收到一封邮件，是确认你的提交，然后就是等待。如果是一个新的包，CRAN 还会运行一些额外的测试，可能比提交包的更新版本要花更多的时间（大约四五天）。

直到 CRAN 回复你，可能会告诉一些潜在的问题，你必须在重新提交你的包之前解决这些问题（并增加一点版本号）——或者你很幸运，你的包立即被接受。

在包被 CRAN 接受后，它将被建立在每个平台上。这有可能会发现更多的错误。等待 48 小时，直到所有包的检查都运行完毕，然后进入你的包页面，点击 CRAN checks 的包 results，检查相关问题，若有必要提交一个更新版本的补丁包。

四. 推广包 (可选)

为了更好地宣传和推广你的包，你可以

- 为你的包编写 vignettes (小册子)

相当于是你为如何具体使用你的包写的博客文章，描述你的包所要解决的问题，然后向用户展示如何解决该问题。执行：

```
use_vignette("Evaluation-Algorithm")    # 或 _, 不能用空格
```

这将自动创建 vignettes/Evaluation-Algorithm.Rmd，向 DESCRIPTION 添加必要的依赖项 (将 knitr 添加到 Suggests 和 VignetteBuilder 字段)。接着，按照标准的 R markdown 格式，编写 Vignette 内容即可。

- 为你的包建立网站

只要你已经遵照了上述流程，在你的 GitHub 仓库里有一个 R 包结构，借助 pkgdown 包，只需要运行：

```
pkgdown::build_site()
```

就能自动把你的包渲染成一个网站，该网站遵循你的包的结构，有一个基于 README 文件的登录页面，一个你的 vignette 折叠页面，以及基于你的 man/文件夹内容的函数引用页面，还有一个专门的 NEWS.md 页面。它甚至包括一个侧边栏，上面有 GitHub 仓库的链接、作者的名字等。

- 为你的包设计六边形 `logo`（目前非常流行），有 `hexSticker` 包和在线网站http://connect.thinkr.fr/hexmake
- 为你的包制作 `cheatsheet`，RStudio 提供有模板

本篇主要参阅 (张敬信, 2022), (Hadley Wickham, 2021), 以及 Cosima Meyer: How to write your own R package and publish it on CRAN，模板感谢 (黄湘云, 2021), (谢益辉, 2021).

参考文献

Hadley Wickham, J. B. (2021). *R Packages*. O' Reilly, 2 edition.

张敬信 (2022). *R* 语言编程：基于 *tidyverse*. 人民邮电出版社, 北京.

谢益辉 (2021). *rmarkdown: Dynamic Documents for R*.

黄湘云 (2021). *Github: R-Markdown-Template*.