

12. R包开发

罗翔宇

中国人民大学统计与大数据研究院

此课件内容基于Hadley Wichham编著的《R包开发》
(O'Reilly, 杨学辉译)

R包简介

- ▶ 在R中，可分享代码的基本单位是包
- ▶ 包把代码、数据、文档和测试整合在一起
- ▶ R包的一个公共发布网站：CRAN(Comprehensive R Archive Network)
- ▶ 从CRAN安装包中： `install.packages("x")`
- ▶ 在R中使用包： `library(x)`
- ▶ 获取包的帮助： `help(package = "x")`
- ▶ 开发R包的好处
 - ▶ 分享代码、方便别人使用
 - ▶ 节省你的时间（只要按照模板即可）
 - ▶ 遵循R包的约定，可以免费获得许多工具

R包简介

► R包开发理念

- 任何可以自动化的都应该自动化，把手动量降到最小
- 目标是让你将时间用于思考你想要包做什么，而不是包的各种细节
- devtools包用来实现此理念，它和Rstudio一起，让你无需关心包是怎么建立的这种低级细节。更多细节，可以查阅官方R扩展开发手册

► 入门(R版本 > 3.1.2)

```
> install.packages(c("devtools", "roxygen2", "testthat", "knitr"))
```

- 约定：foo()指函数，bar指变量或函数参数，baz/指路径

► 版本记录：

```
> devtools::session_info()
- session info -----
setting  value
version  R version 3.6.2 (2019-12-12)
os       windows >= 8 x64
```

给包命名

► 命名的要求

- 名称只能包含字母、数字和点号(.), 必须以一个字母开始, 且不能以点号结束
- 不能在包名中使用连字符(-)或下划线(_)

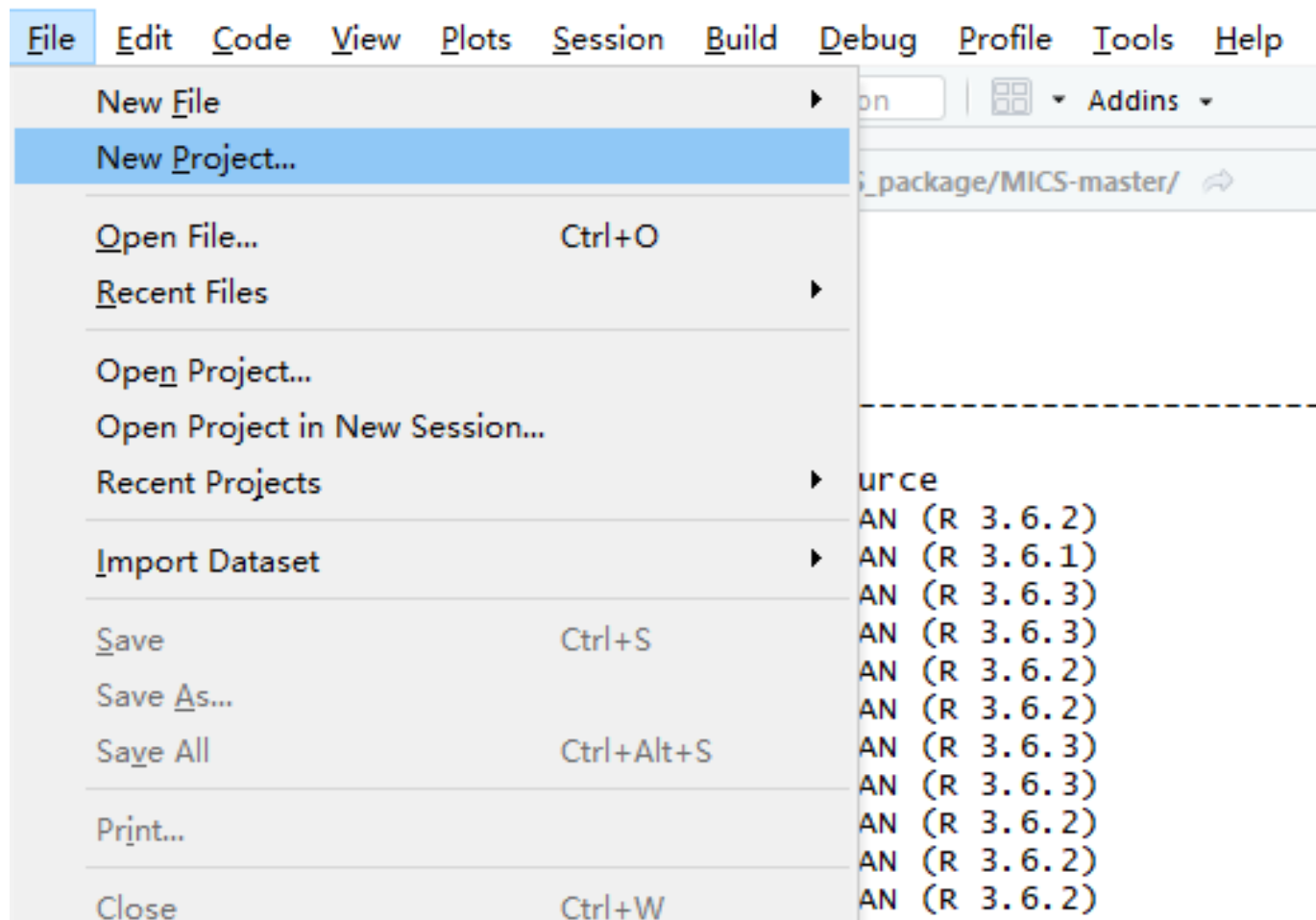
► 命名的策略

- 选择一个很容易被搜索到的独特名字。可以在CRAN加载 [http://cran.r-project.org/web/packages/\[包名\]](http://cran.r-project.org/web/packages/[包名]) 来检查一个名称是否已经被使用
- 避免大写和小写交叉, 比如RgtK2, RGtK2
- 找到一个和这个问题相关的词, 修改它, 使它特别
 - plyr是apply函数族的推广, 并使人想到工具钳子
- 使用缩略词
 - Rcpp = R + C++

创建一个包

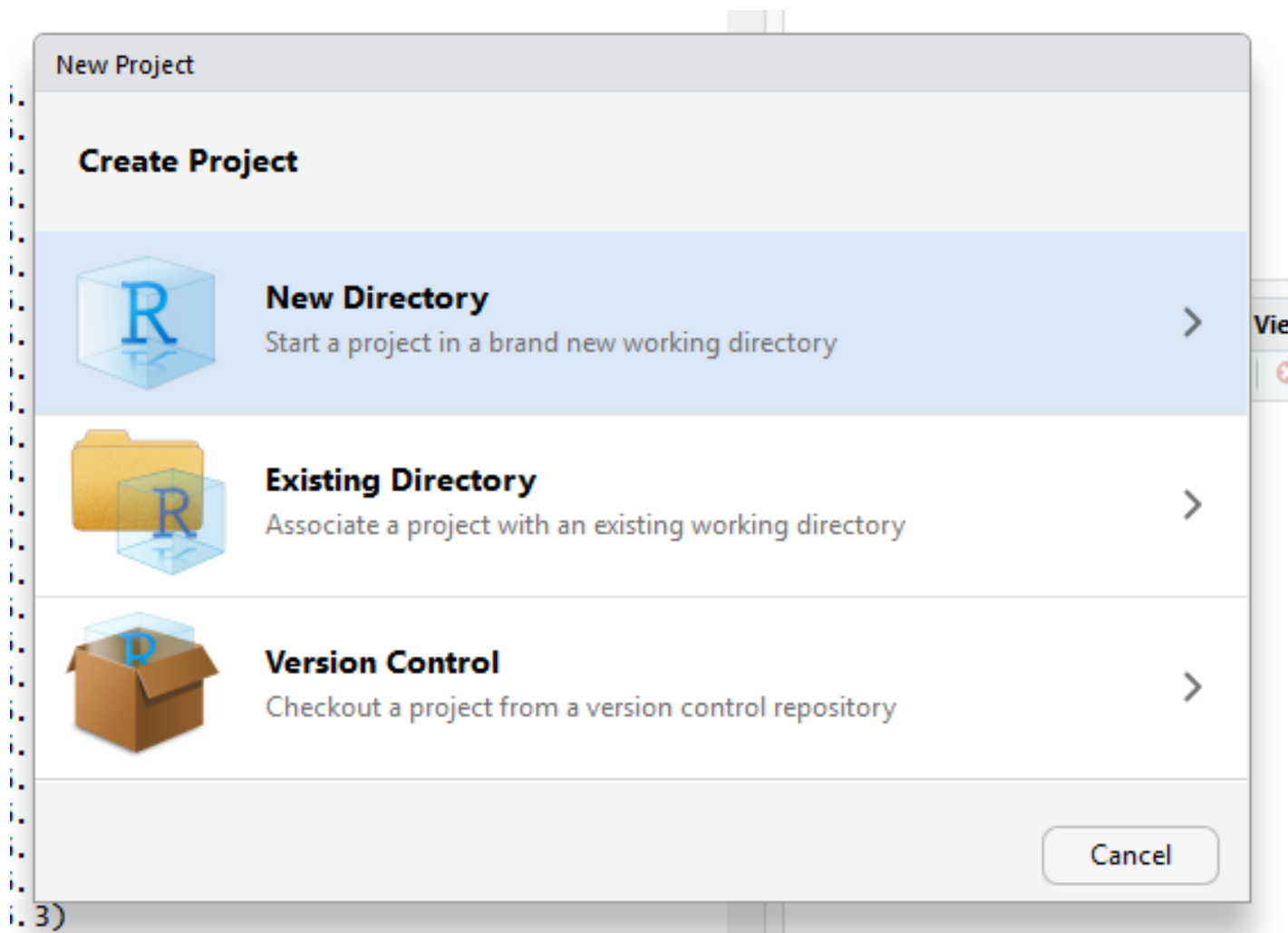
一旦确定了包的名字，有两种方法来创建包

► (1) 单击File|New Project



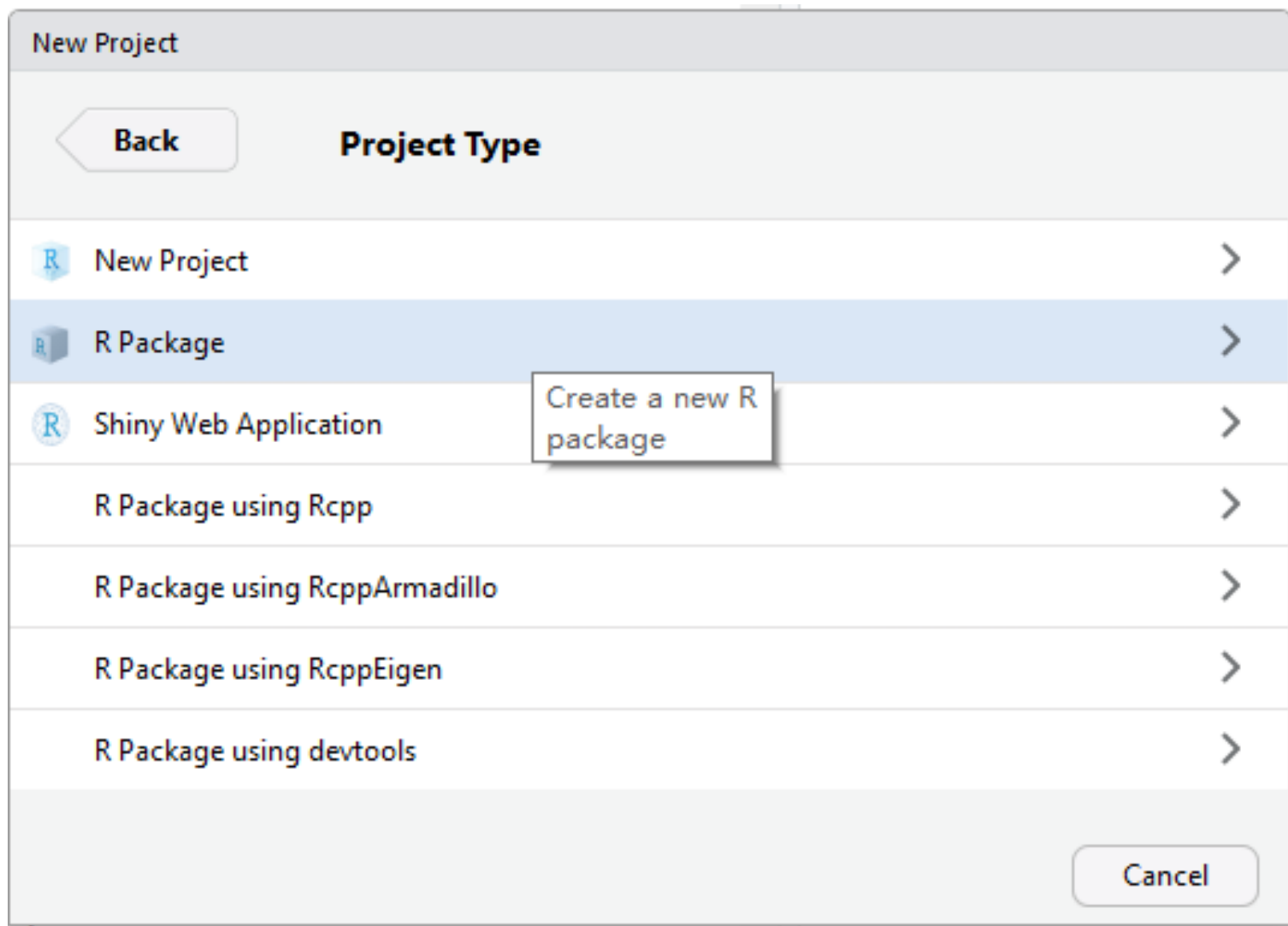
创建一个包

► (2) 选择New Directory



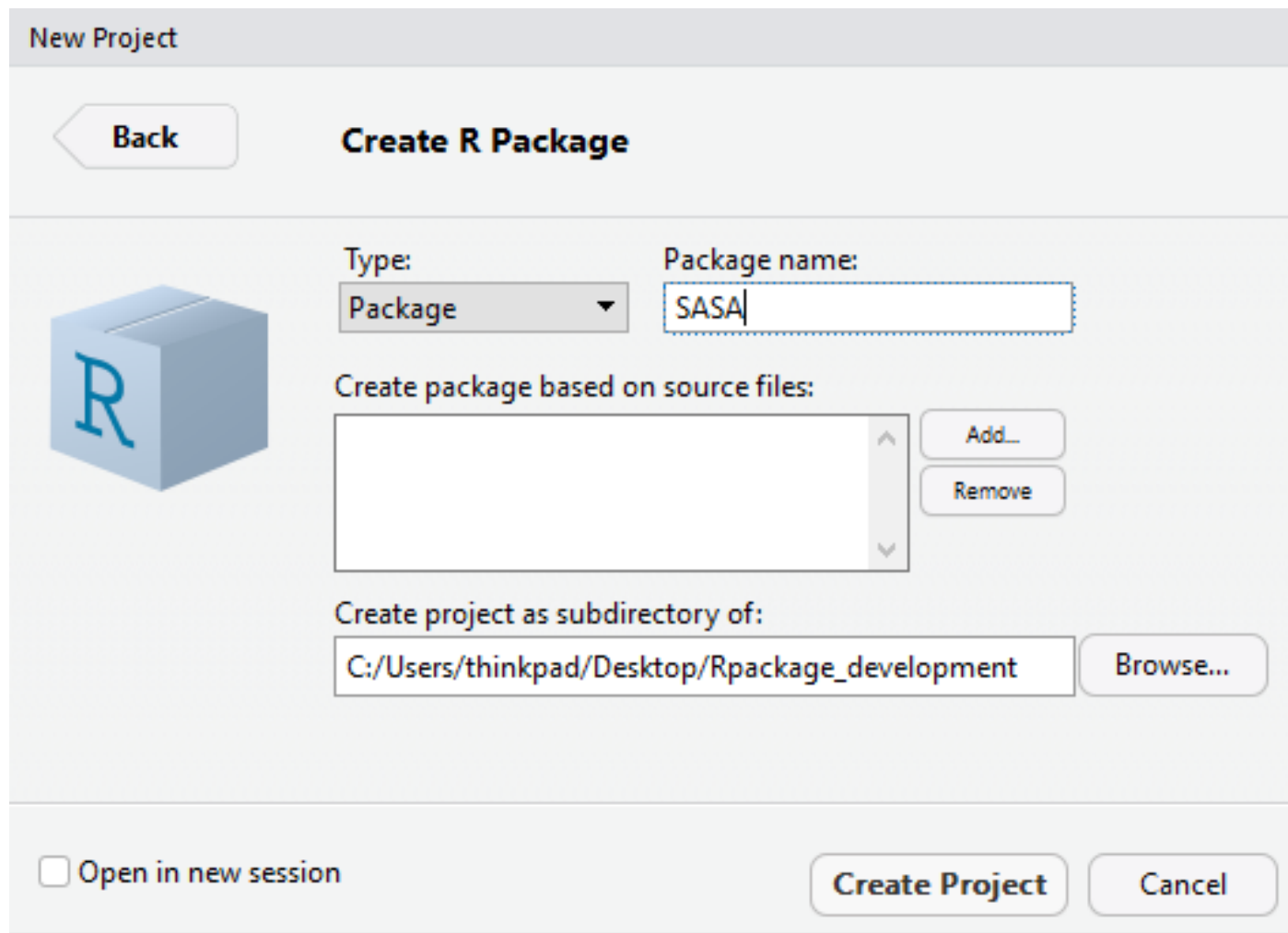
创建一个包

► (3) 选择R package



创建一个包

- (4) 给出你的包名，然后单击Create Project



The screenshot shows the 'New Project' dialog box in R Studio, specifically the 'Create R Package' step. The dialog has a title bar 'New Project' and a 'Back' button. The main title is 'Create R Package'. On the left is the R logo. The 'Type' dropdown is set to 'Package'. The 'Package name' text box contains 'SASA'. Below this is a section 'Create package based on source files:' with an empty list box and 'Add...' and 'Remove' buttons. The 'Create project as subdirectory of:' text box contains 'C:/Users/thinkpad/Desktop/Rpackage_development', with a 'Browse...' button next to it. At the bottom, there is an unchecked checkbox 'Open in new session' and two buttons: 'Create Project' and 'Cancel'.

New Project

Back

Create R Package

Type: Package

Package name: SASA

Create package based on source files:

Add...

Remove

Create project as subdirectory of:

C:/Users/thinkpad/Desktop/Rpackage_development

Browse...

☐ Open in new session

Create Project

Cancel

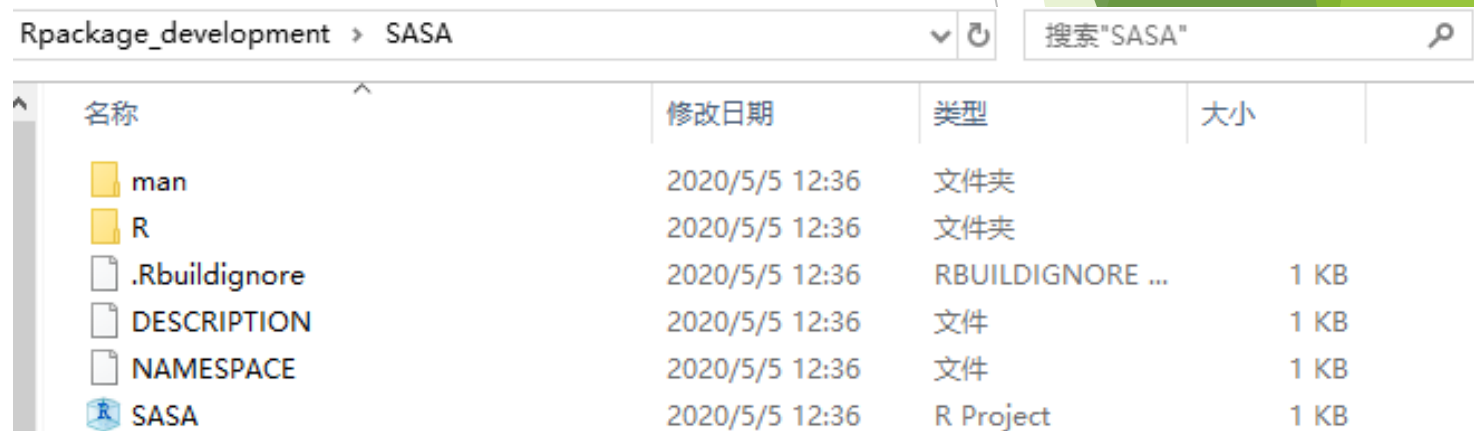
创建一个包

- ▶ 第二种方法则是通过R中的命令来创建一个新的包

```
> devtools::create("path/to/package/pkgname")
```

- ▶ 这两种方法得到的结果都是一样的，一个最小的可用包，它有三个组成部分

- ▶ 一个R/ 目录
- ▶ 一个描述文件DESCRIPTION
- ▶ 一个命名空间NAMESPACE



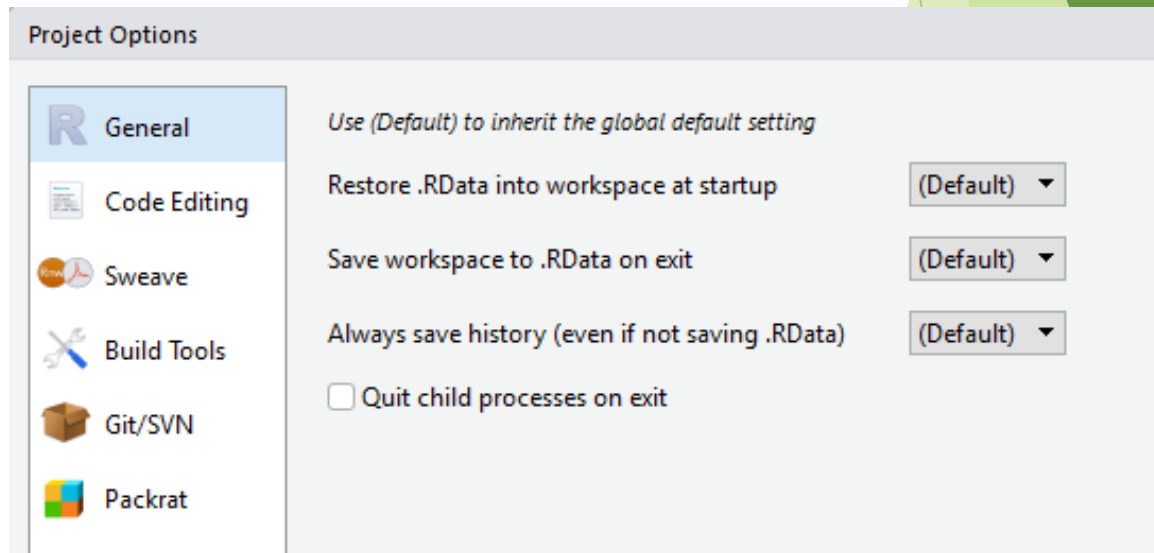
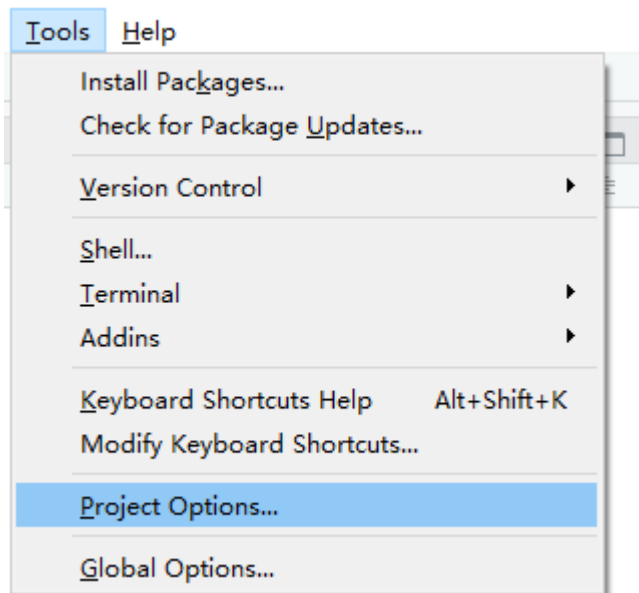
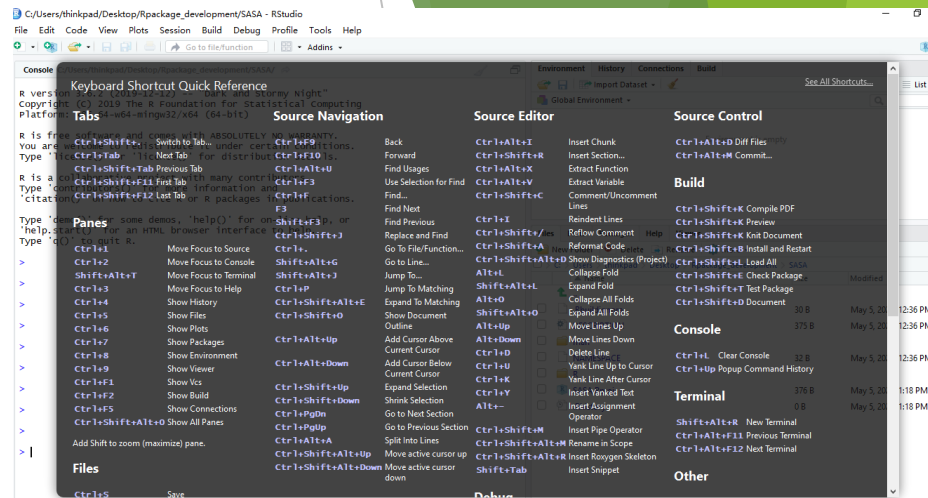
The screenshot shows a file explorer window with the path 'Rpackage_development > SASA'. The search bar contains '搜索"SASA"'. The table below lists the files and folders in the directory.

名称	修改日期	类型	大小
man	2020/5/5 12:36	文件夹	
R	2020/5/5 12:36	文件夹	
.Rbuildignore	2020/5/5 12:36	RBUILDIGNORE ...	1 KB
DESCRIPTION	2020/5/5 12:36	文件	1 KB
NAMESPACE	2020/5/5 12:36	文件	1 KB
SASA	2020/5/5 12:36	R Project	1 KB

- ▶ 通过Rstudio还包括一个项目文件，使你的包易于在Rstudio中使用
- ▶ 建议不使用package.skeleton()创建包，会做一些额外的工作才能获得一个可用的包

RStudio项目

- ▶ 双击生成的Rproj文件，你可以使用RStudio的图形用户界面或者命令行工具
- ▶ 项目是开发包的一个很好方式
 - ▶ 每个项目都是独立的，在一个项目中运行的代码不会影响任何其它项目
 - ▶ 获得一些有用的快捷键，可以通过Alt+Shift+K查询
- ▶ 对项目文件进行调整的选项



什么是包

- ▶ 包生命周起的五个状态：源码包、压缩包、二进制包、已安装的包和内存中的包
- ▶ 源码包：你电脑上那个处于开发版本的包。源码包只是包含R/子目录，DESCRIPTION等组件的一个目录
- ▶ 压缩包：它是一个已经压缩为单个文件的包，R中的压缩包使用.tar.gz扩展名。解压后和源码包的主要区别
 - ▶ 使用指南被创建
 - ▶ 你的源码包中可能包含开发时产生的一些临时文件
 - ▶ .Rbuildignore文件中列出的任何文件都不会出现在解压的包中
- ▶ 二进制包：向没有包开发工具的R用户发布包。解压后的不同
 - ▶ R/目录中没有.R文件，但有一些其它用于高效的文件格式来存储解析后的函数
 - ▶ Meta/目录包含大量Rds文件，html/目录包含HTML帮助文件

什么是包

- ▶ 已安装的包：是解压到一个包库的二进制包
- ▶ 支持所有包安装的工具是R的命令行工具R CMD INSTALL，它可以安装源码包、压缩包、二进制包
- ▶ `install.packages()`用于下载和安装CRAN编译的二进制包
- ▶ `install_github()`的工作方式稍有不同，它从GitHub上下载源码包、编译、然后安装
- ▶ 内存中的包：要使用包，必须把它加载到内存中。`library()`

什么是库

► 什么是库？

- 库是一个包含已安装包的目录
 - 每个人都至少有两个库：一个用来放已经安装的包，另一个用来放R安装时自带的包(如base, stats等)
 - 用户安装包的目录在不同的R版本下有所不同，重新安装R以后，它们似乎不见了，但是它仍然在硬盘中，只是R找不到而已
 - 可以通过`.libPaths()`来查看正在使用的库是哪个
- library()和require()的主要区别是：在包找不到的情况下，library()抛出一个错误，require()则打印一个消息并返回FALSE

```
> .libPaths()
[1] "C:/Users/thinkpad/Documents/R/win-library/3.6"
[2] "C:/Program Files/R/R-3.6.2/library"

> library("superman")
Error in library("superman") : 不存在叫'superman'这个名字的程辑包
> require("superman")
载入需要的程辑包：superman
warning message:
In library(package, lib.loc = lib.loc, character.only = TRUE, logical.return = TRUE, :
  不存在叫'superman'这个名字的程辑包
```

包的组件—R代码

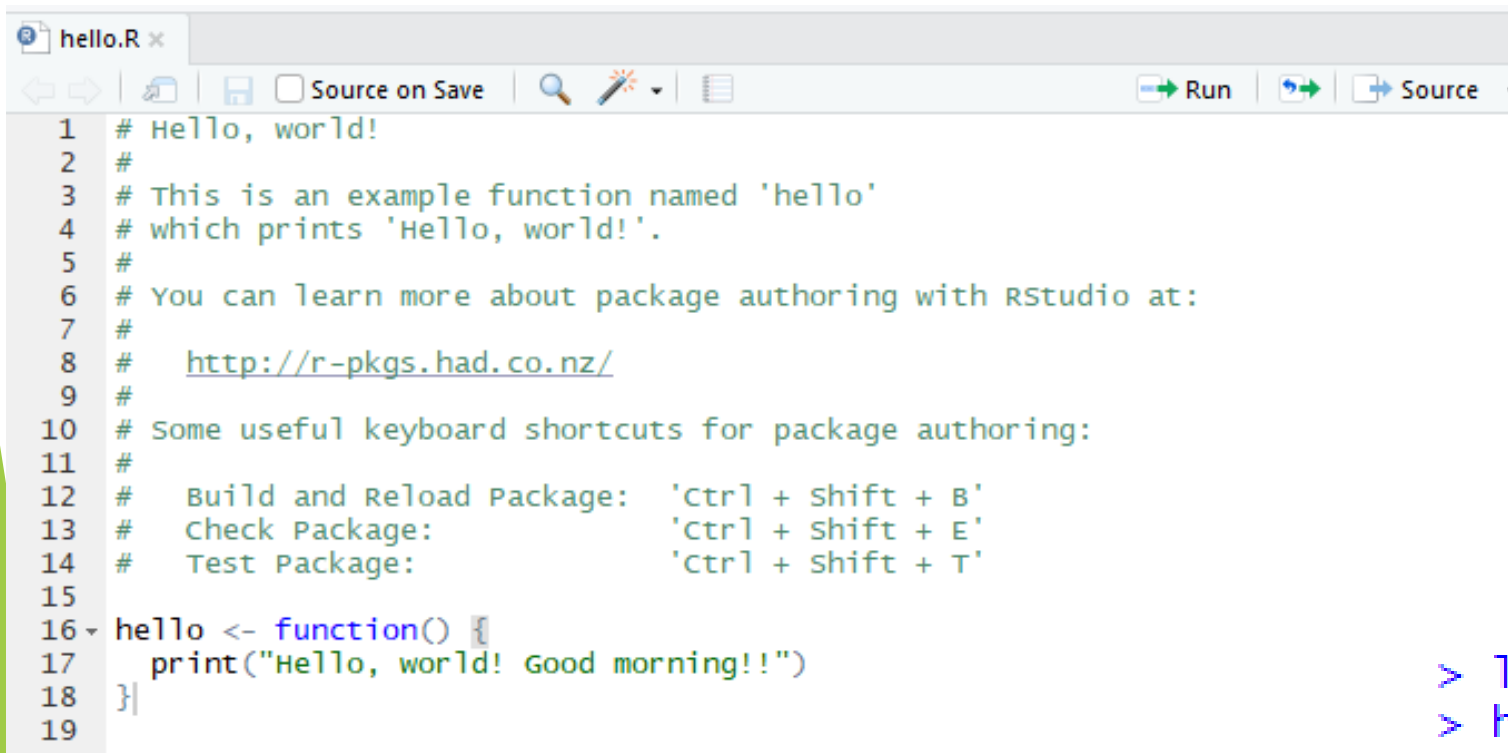
- ▶ 使用包的第一个原则是，所有的R代码都要放在R/目录中
- ▶ 当修改了R代码后，很容易重新加载你的代码
- ▶ 旧代码以及输出

```
hello.R x
Source on Save Run
1 # Hello, world!
2 #
3 # This is an example function named 'hello'
4 # which prints 'Hello, world!'.
5 #
6 # You can learn more about package authoring with RStudio at:
7 #
8 # http://r-pkgs.had.co.nz/
9 #
10 # Some useful keyboard shortcuts for package authoring:
11 #
12 #   Build and Reload Package: 'Ctrl + Shift + B'
13 #   Check Package:           'Ctrl + Shift + E'
14 #   Test Package:            'Ctrl + Shift + T'
15
16 hello <- function() {
17   print("Hello, world!")
18 }
19
```

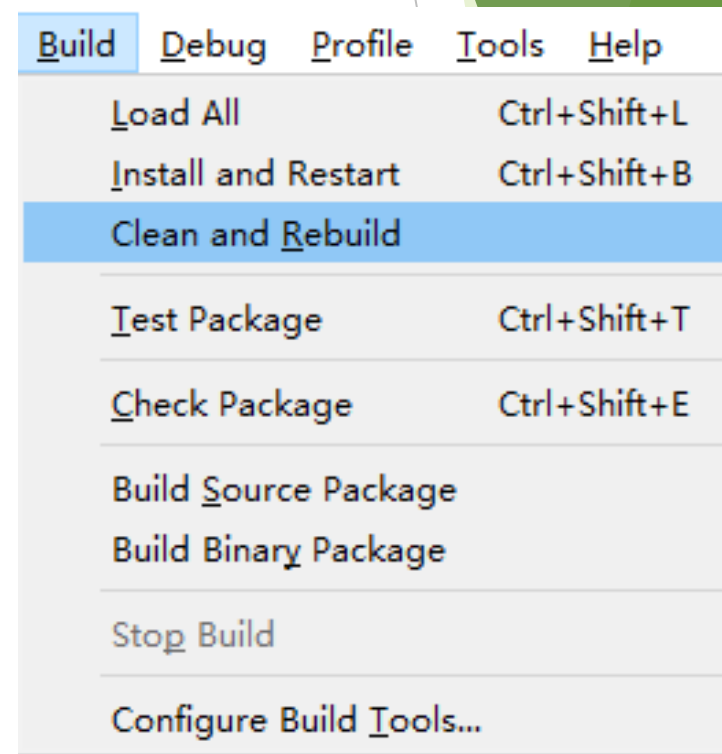
```
> library(SASA)
> hello()
[1] "Hello, world!"
```

包的组件—R代码

- ▶ 使用包的第一个原则是，所有的R代码都要放在R/目录中
- ▶ 当修改了R代码后，很容易重新加载你的代码
- ▶ 新代码、重新加载、以及输出



```
1 # Hello, world!
2 #
3 # This is an example function named 'hello'
4 # which prints 'Hello, world!'.
5 #
6 # You can learn more about package authoring with RStudio at:
7 #
8 # http://r-pkgs.had.co.nz/
9 #
10 # Some useful keyboard shortcuts for package authoring:
11 #
12 #   Build and Reload Package: 'Ctrl + Shift + B'
13 #   Check Package:           'Ctrl + Shift + E'
14 #   Test Package:            'Ctrl + Shift + T'
15
16 hello <- function() {
17   print("Hello, world! Good morning!!")
18 }
19
```



```
> library(SASA)
> hello()
[1] "Hello, world! Good morning!!"
```

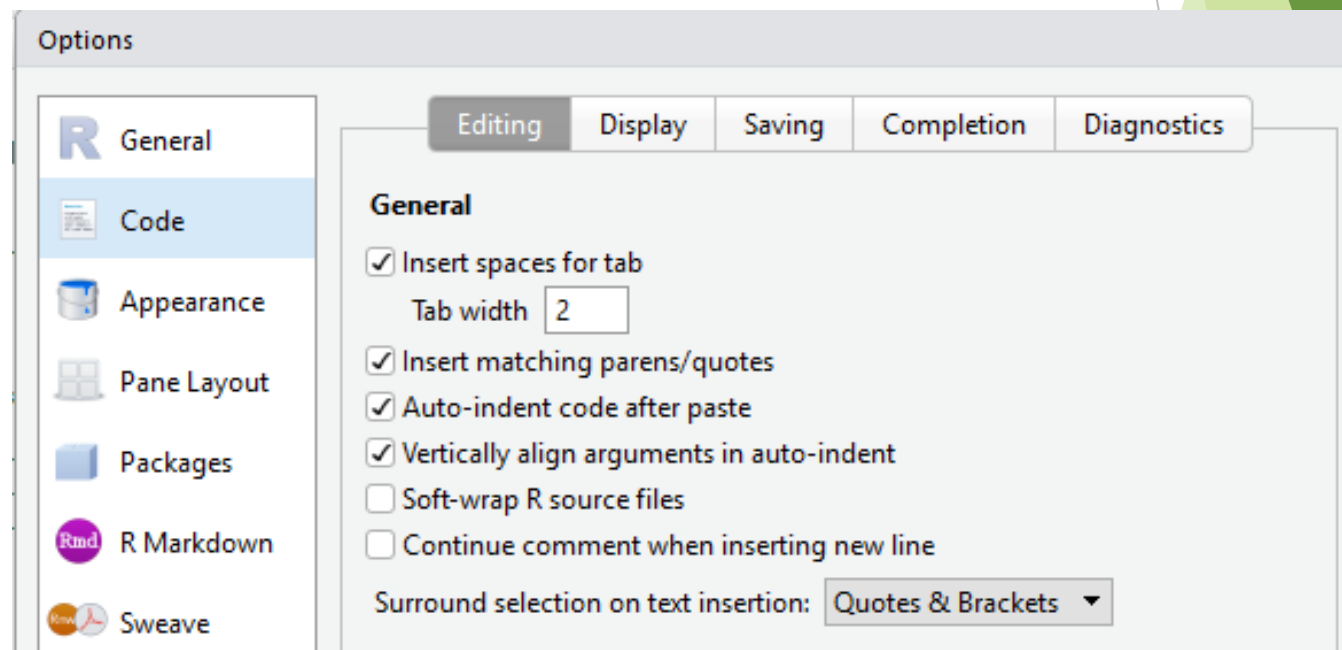
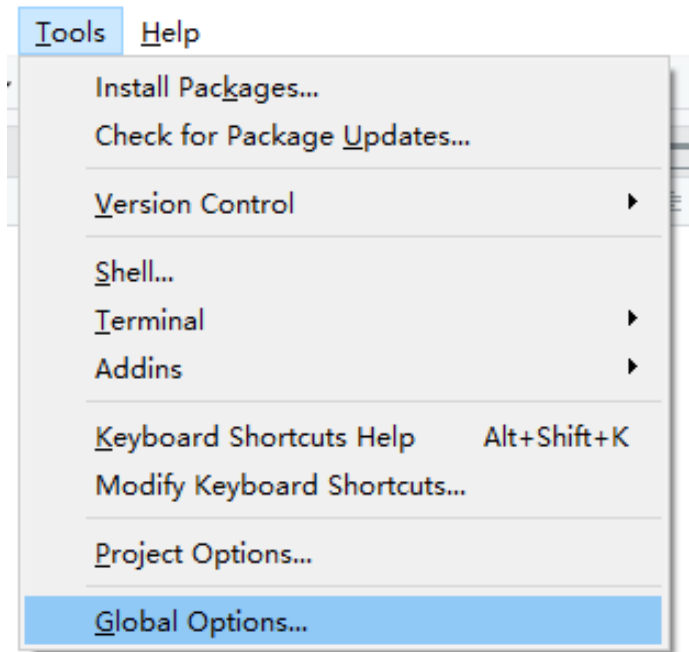

包的组件—R代码

- ▶ 组织函数：当编写函数时，两种极端做法是很不好的
 - ▶ 把所有函数都放在一个文件中
 - ▶ 每个函数都放在独立的文件中
- ▶ 文件名：
 - ▶ 好的，fit_models.R, utility_functions.R;
 - ▶ 不好的，foo.r, stuff.r
- ▶ 代码风格：
 - ▶ 对象名称：变量名和函数名应该小写。使用下划线(_)分割名字内的单词，或者使用驼峰式大小写(YouAre)，但是请保持一致
 - ▶ 避免使用现有函数名和变量名(T, c, mean)
 - ▶ 空格：在所有的中缀运算符(=, +, -, <-等)前后使用空格；不要再:, ::和:::前后使用空格；不要在圆括号或中括号中的代码前后使用空格

包的组件—R代码

► 代码风格：

- 花括号：左花括号不要另起一行，并且它的后面总是跟着一个换行符。右花括号总是独占一行，除非它的后面跟着else。
- 行的长度：每行代码限定在80个字符内
- 缩进：使用两个空格缩进你的代码，不要使用Tab或混合使用Tab和空格，你可以通过偏好对话框中修改这些选项



包的组件—R代码

► 代码风格:

► 缩进: 唯一的例外是跨越多行的函数定义

```
long_function_name <- function(a = "a long argument",  
                                b = "another argument",  
                                c = "another long argument"){  
  ##  
}
```

► 赋值表达式: 使用 <- 不要使用 =

► 注释指南: 注释的每一行应以注释符号加单个空格开头; 在注释行中使用-或=把文件分给成容易阅读的代码块

```
# load data -----
```

```
# plot data =====
```

包的元数据

- DESCRIPTION描述文件的作用是存储包中重要的元数据

```
Package: SASA
Type: Package
Title: What the Package Does (Title Case)
Version: 0.1.0
Author: Who wrote it
Maintainer: The package maintainer <yourself@somewhere.net>
Description: More about what it does (maybe more than one line)
    Use four spaces when indenting paragraphs within the Description.
License: What license is it under?
Encoding: UTF-8
LazyData: true
```

- 依赖：包需要什么

- 需要列出该包所依赖的包。以下行表明包需要ggvis和dplyr来工作
- Imports: dplyr, ggvis
- Imports列表里的包将被必须安装
- 下面行表明虽然包可以使用ggvis和dplyr，但它们不是必需的
- Suggests: dplyr, ggvis

包的元数据

► 依赖：包需要什么

- 版本。如果需要特定版本的包，则在包后面的括号中指定它
- Imports: `ggvis (>= 0.2)`, `dplyr (>= 0.3.0.1)`
- 版本控制是发布包时最重要的事情，如果你在描述文件中提供了版本号，用户(他的某个包可能过时)将会收到明确的错误消息：此包过时
- 其他依赖：
 - Depends: 在R 2.14.0中的命令空间推出之前，Depends是唯一依赖于另一个包的方法。现在应该几乎使用Imports而不是Depends。可以使用Depends来指定一个特定的R版本，例如`Depends: R(>=3.0.1)`
 - LinkingTo: 列出的包依赖于另一个包中的C或者C++代码

包的元数据

► 标题和描述：包是做什么的

- Title是包的一行描述，经常显示在包列表中，保持简短
- Description比标题更详细，可以使用多个句子，但只限于一段
- 也可以利用README.md文件进行更深入的描述
- 我开发的一个R包的例子：

BUScorrect

platforms all rank 1660 / 1904 posts 0 in Bioc 1.5 years
build ok updated before release dependencies 30

DOI: [10.18129/B9.bioc.BUScorrect](https://doi.org/10.18129/B9.bioc.BUScorrect)  

Batch Effects Correction with Unknown Subtypes

Bioconductor version: Release (3.11)

High-throughput experimental data are accumulating exponentially in public databases. However, mining valid scientific discoveries from these abundant resources is hampered by technical artifacts and inherent biological heterogeneity. The former are usually termed "batch effects," and the latter is often modelled by "subtypes." The R package BUScorrect fits a Bayesian hierarchical model, the Batch-effects-correction-with-Unknown-Subtypes model (BUS), to correct batch effects in the presence of unknown subtypes. BUS is capable of (a) correcting batch effects explicitly, (b) grouping samples that share similar characteristics into subtypes, (c) identifying features that distinguish subtypes, and (d) enjoying a linear-order computation complexity.

Author: Xiangyu Luo <xyluo1991 at gmail.com>, Yingying Wei <yweicuhk at gmail.com>

Maintainer: Xiangyu Luo <xyluo1991 at gmail.com>

包的元数据

► 作者：你是谁？

Author: Who wrote it

Maintainer: The package maintainer <yourself@somewhere.net>

► 许可证(License): 谁能使用包

- MIT: 它让人们使用和自由分发你的代码，但是有一个限制：许可证必须始终和代码一起分发
- GPL-2或者GPL-3: 任何包含你的代码的包都必须使用GPL兼容的许可证来发布。任何人发布你代码的修改版本时，必须公布源码
- CC0: 这个许可证放弃了你对代码和数据的所有权利，任何人都可以自由地把它用于任何目的。
- 如果想了解关于其他常见许可证的更多内容，GitHub的 [chooselicense.com](https://choosealicense.com) 可以了解更多

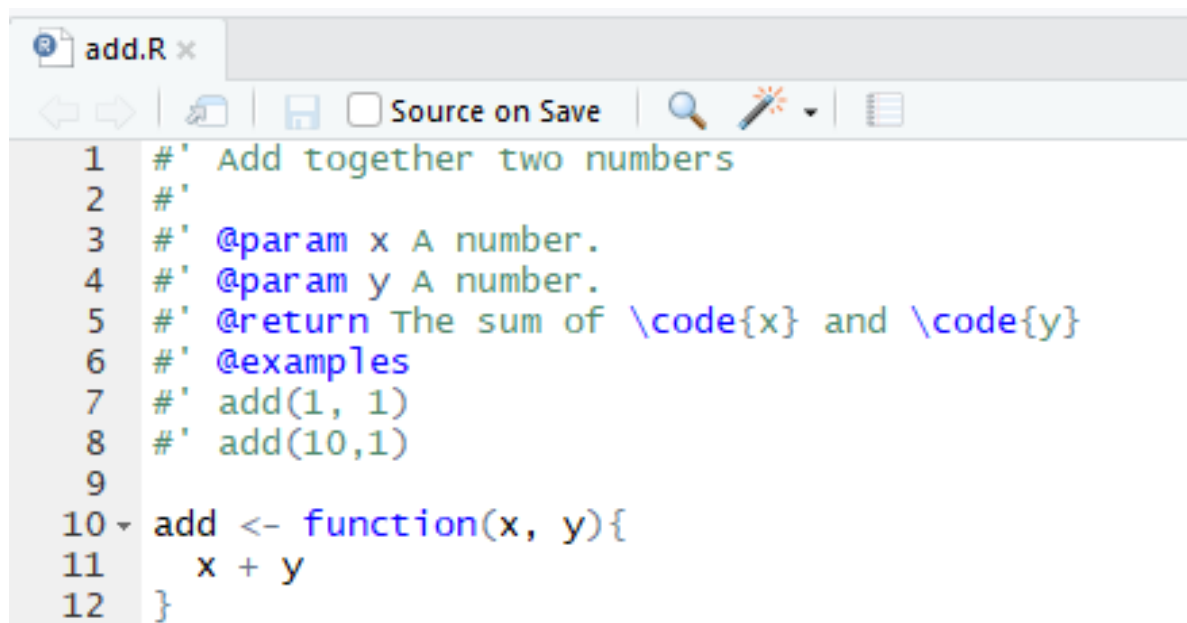
对象文档

- ▶ 没有文档，用户将不知道如何使用包
- ▶ 对象文档是一种引用类型的文件。它就像字典，如果想知道一个词(函数)的意思，可以通过?或者help()查询
- ▶ R提供了一个标准的方法来给包中的对象提供文档：在man/目录下提供一些.Rd文件
- ▶ 我们不用手工写这些文件，而是用roxygen2包，它将特殊格式的注释转为.Rd文件。roxygen2的目标是使代码文档尽可能好写，优点为
 - ▶ 代码和文档混合在一起，当修改代码时，可以顺手修改文档
 - ▶ roxygen2会动态检查需要提供文档的对象，自动生成文档模板

对象文档

► 文档工作流程

- 添加roxygen注释到.R文件
- 将roxygen注释转为.Rd文件
- 利用?预览文档
- 修改注释，重复上面步骤，直到文档看起来是你想要的样子










The screenshot shows an R script editor window titled 'add.R'. The script contains roxygen2 comments for a function named 'add'. The comments are as follows:

```
1 #' Add together two numbers
2 #'
3 #' @param x A number.
4 #' @param y A number.
5 #' @return The sum of \code{x} and \code{y}
6 #' @examples
7 #' add(1, 1)
8 #' add(10,1)
9
10 add <- function(x, y){
11   x + y
12 }
```

对象文档

```
> devtools::document()  
Updating SASA documentation  
First time using roxygen2. Upgrading automatically...  
Loading SASA  
Warning: The existing 'NAMESPACE' file was not generated by roxygen2, and will not be over  
erwritten.  
Writing add.Rd
```

名称	修改日期	类型	大小
 man	2020/5/5 15:54	文件夹	
 R	2020/5/5 15:54	文件夹	
 .Rbuildignore	2020/5/5 12:36	RBUILDIGNORE ...	1 KB
 .Rhistory	2020/5/5 13:26	RHISTORY 文件	0 KB
 DESCRIPTION	2020/5/5 15:52	文件	1 KB
 NAMESPACE	2020/5/5 12:36	文件	1 KB
 SASA	2020/5/5 14:08	R Project	1 KB

对象文档

```
add.Rd
1  % Generated by roxygen2: do not edit by hand
2  % Please edit documentation in R/add.R
3  \name{add}
4  \alias{add}
5  \title{Add together two numbers}
6  \usage{
7    add(x, y)
8  }
9  \arguments{
10 \item{x}{A number.}
11
12 \item{y}{A number.}
13 }
14 \value{
15 The sum of \code{x} and \code{y}
16 }
17 \description{
18 Add together two numbers
19 }
20 \examples{
21 add(1, 1)
22 add(10,1)
23 }
24
```

对象文档

- ▶ 通过在RStudio中输入命令行 `?add`，你可以看到对应函数的帮助文档

`add {SASA}`

R Documentation

Add together two numbers

Description

Add together two numbers

Usage

`add(x, y)`

Arguments

x A number.

y A number.

Value

The sum of **x** and **y**

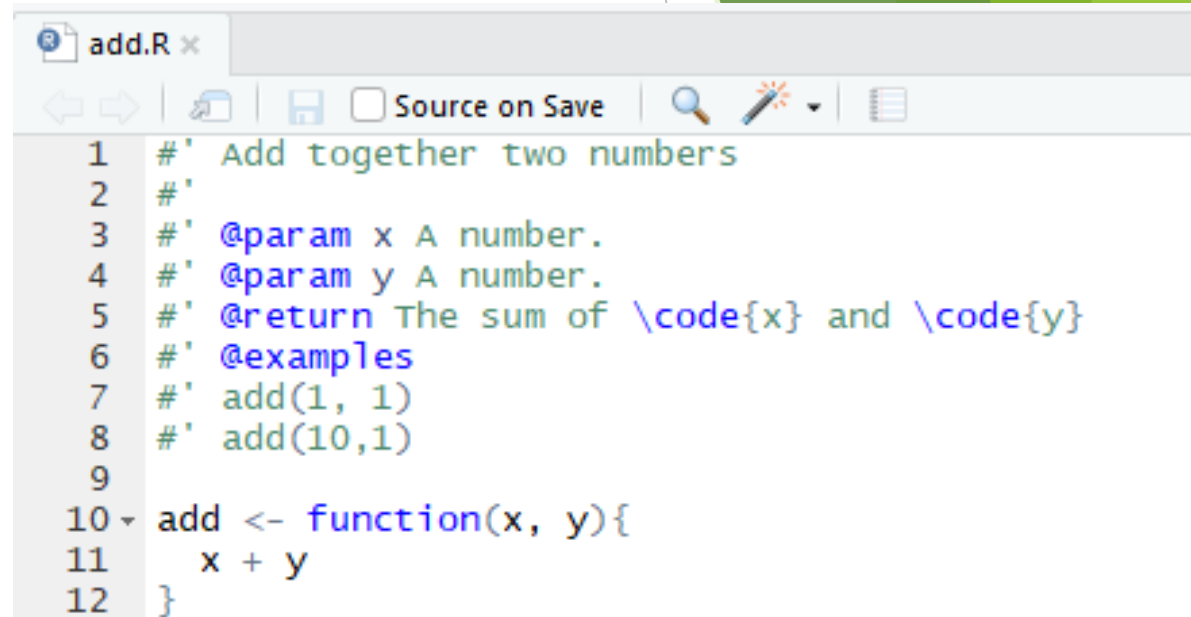
Examples

`add(1, 1)`

`add(10, 1)`

对象文档

- ▶ roxygen注释以 '#' 开始，放在函数的前面。一个函数前所有的roxygen行被称为一个块(block)
- ▶ 第一句是文档的标题
- ▶ @param 名字 描述
 - ▶ 此标签描述了函数的输入或参数
 - ▶ 应该简洁地描述参数的类型，该参数干什么的
 - ▶ 描述应以大写字母开头、以句号结尾
 - ▶ 所有的参数都必须提供文档



```
add.R x
1 #' Add together two numbers
2 #'
3 #' @param x A number.
4 #' @param y A number.
5 #' @return The sum of \code{x} and \code{y}
6 #' @examples
7 #' add(1, 1)
8 #' add(10,1)
9
10 add <- function(x, y){
11   x + y
12 }
```

对象文档

► @examples

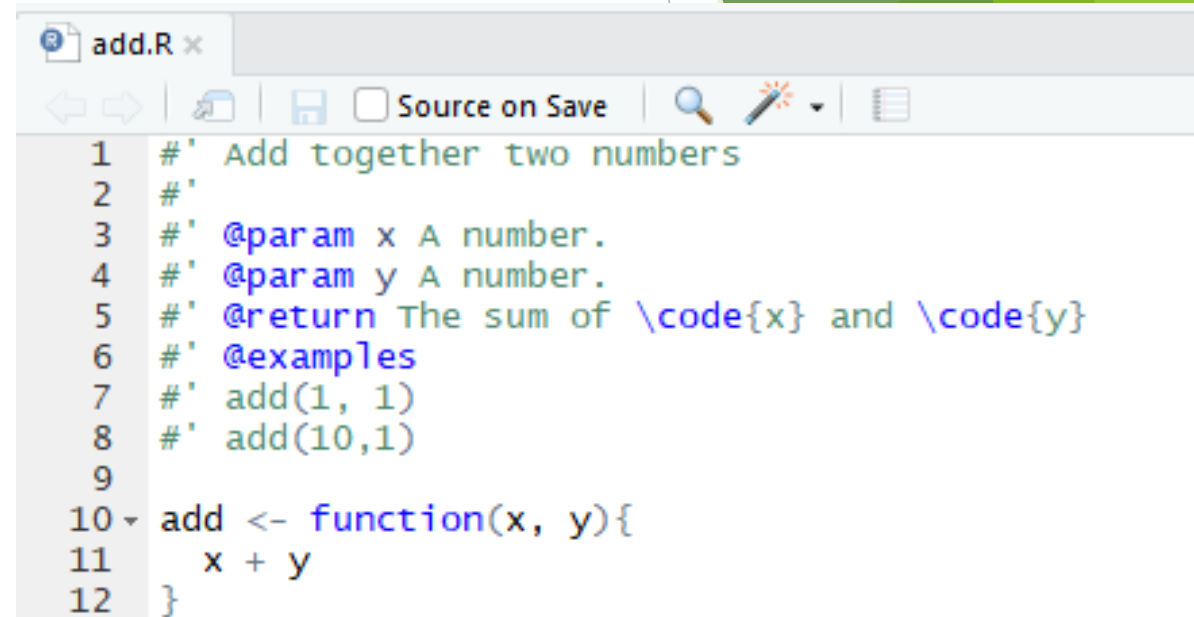
- 这个标签提供了可执行的R代码，演示了如何在实际例子中运行该函数
- 这是文档中非常重要的一部分，因为很多人先看例子

► @return 返回值描述

- 这个标签描述了函数的输出

► 特殊字符

- @ 使用@@将会在最终的文档中插入@
- % 使用\%可在文档中插入%
- \ 使用\\可在文档中插入\



```
add.R x
1 #' Add together two numbers
2 #'
3 #' @param x A number.
4 #' @param y A number.
5 #' @return The sum of \code{x} and \code{y}
6 #' @examples
7 #' add(1, 1)
8 #' add(10,1)
9
10 add <- function(x, y){
11   x + y
12 }
```

对象文档

► 文本格式参考

► `\emph{}` 斜体

► `\strong{}` 粗体

► `\code{}` R代码形式

► 到其他文件的链接: `\code{\link{function}}`

► 到网络的链接: `\url{http://rstudio.com}`

► 列表 #' `\enumerate{`

 #' `\item first project`

 #' `\item second project`

 #' }

► 数学符号: `\eqn{a + b}` 行内方程, `\deqn{a + b}` 块方程

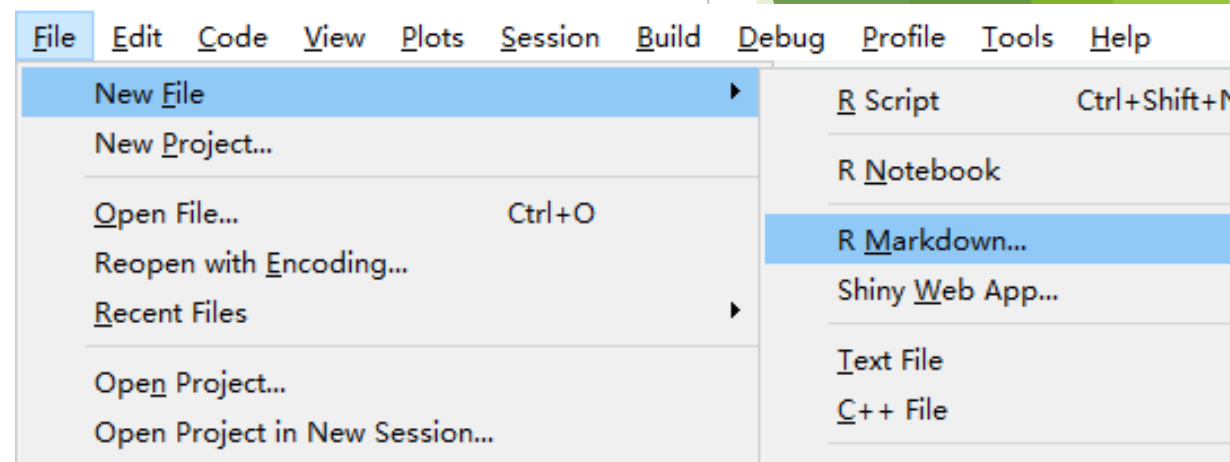
使用指南：长篇文档


- ▶ 使用指南是包的长篇指南。使用指南就像是一本书的一章或一篇学术论文：它可以说明包想要解决的问题，亦可以用来解释包中的细节
- ▶ 许多现有的包都有使用指南，可以通过 `browseVignettes("pkgname")` 来进行查看
- ▶ 在R3.0.0之前，创建使用指南的唯一方法是用Sweave,但它只和LaTeX一起工作
- ▶ 我们将使用knitr提供的R Markdown使用指南引擎
 - ▶ 用Markdown书写
 - ▶ 可以混合文本、代码、结果

使用指南：长篇文档

- 创建vignettes文件夹，添加my-vignette R markdown文件（可以通过Rstudio创建）

名称	修改日期	类型	大小
man	2020/5/5 15:54	文件夹	
R	2020/5/5 15:54	文件夹	
vignettes	2020/5/5 16:44	文件夹	
.Rbuildignore	2020/5/5 12:36	RBUILDIGNORE ...	1 KB
.Rhistory	2020/5/5 13:26	RHISTORY 文件	0 KB
DESCRIPTION	2020/5/5 15:52	文件	1 KB
NAMESPACE	2020/5/5 12:36	文件	1 KB
SASA	2020/5/5 14:08	R Project	1 KB



 my-vignettes

2020/5/5 16:44

RMD 文件

2 KB

使用指南：长篇文档

- 在Rstudio中编辑my-vignettes文件（代码来自于<https://github.com/yihui/knitr-examples/blob/master/001-minimal.Rmd>）

```
# A minimal R Markdown example
```

A quote:

```
> Markdown is not LaTeX.  
To compile me, run this in R:
```

```
library(knitr)  
knit('001-minimal.Rmd')
```

See [output here](<https://github.com/yihui/knitr-examples/blob/master/001-minimal.md>).

```
## code chunks
```

A `_paragraph_` here. A code chunk below (remember the three backticks):

```
```{r}  
1+1
.4-.7+.3 # what? it is not zero!
```
```

```
## graphics
```

It is easy.

```
```{r}  
plot(1:10)
hist(rnorm(1000))
```
```

```
## inline code
```

Yes I know the value of pi is ``r pi``, and 2 times pi is ``r 2*pi``.

使用指南：长篇文档

```
## math
```

sigh. You cannot live without math equations. OK, here we go:
`\alpha+\beta=\gamma`. Note this is not supported by native markdown. You probably want to try RStudio, or at least the R package **markdown**, or the function ``knitr::knit2html()``.

```
## nested code chunks
```

You can write code within other elements, e.g. a list

1. foo is good

```
```{r}
strsplit('hello indented world', ' ')[[1]]
```
```

2. bar is better

or inside blockquotes:

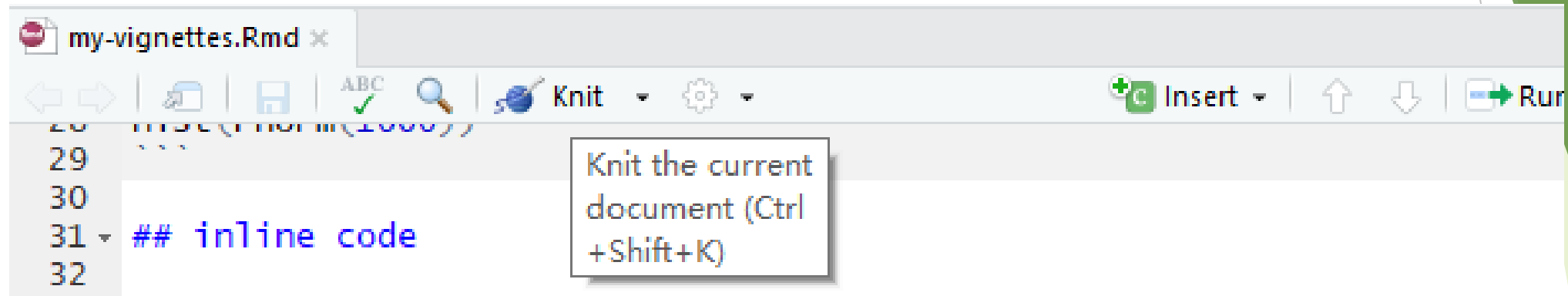
> Here is a quote, followed by a code chunk:

```
>
> ```{r}
> x = 1:10
> rev(x^2)
> ```
```

```
## conclusion
```

Nothing fancy. You are ready to go. When you become picky, go to the [\[knitr website\]](http://yihui.org/knitr/) (<http://yihui.org/knitr/>).

使用指南：长篇文档



my-vignettes

2020/5/5 16:47

HTML 文档

752 KB

使用指南：长篇文档

A minimal R Markdown example

A quote:

Markdown is not LaTeX. To compile me, run this in R:

```
library(knitr)
knit('001-minimal.Rmd')
```

See [output here](#).

code chunks

A *paragraph* here. A code chunk below (remember the three backticks):

```
1+1
```

```
## [1] 2
```

```
.4-.7+.3 # what? it is not zero!
```

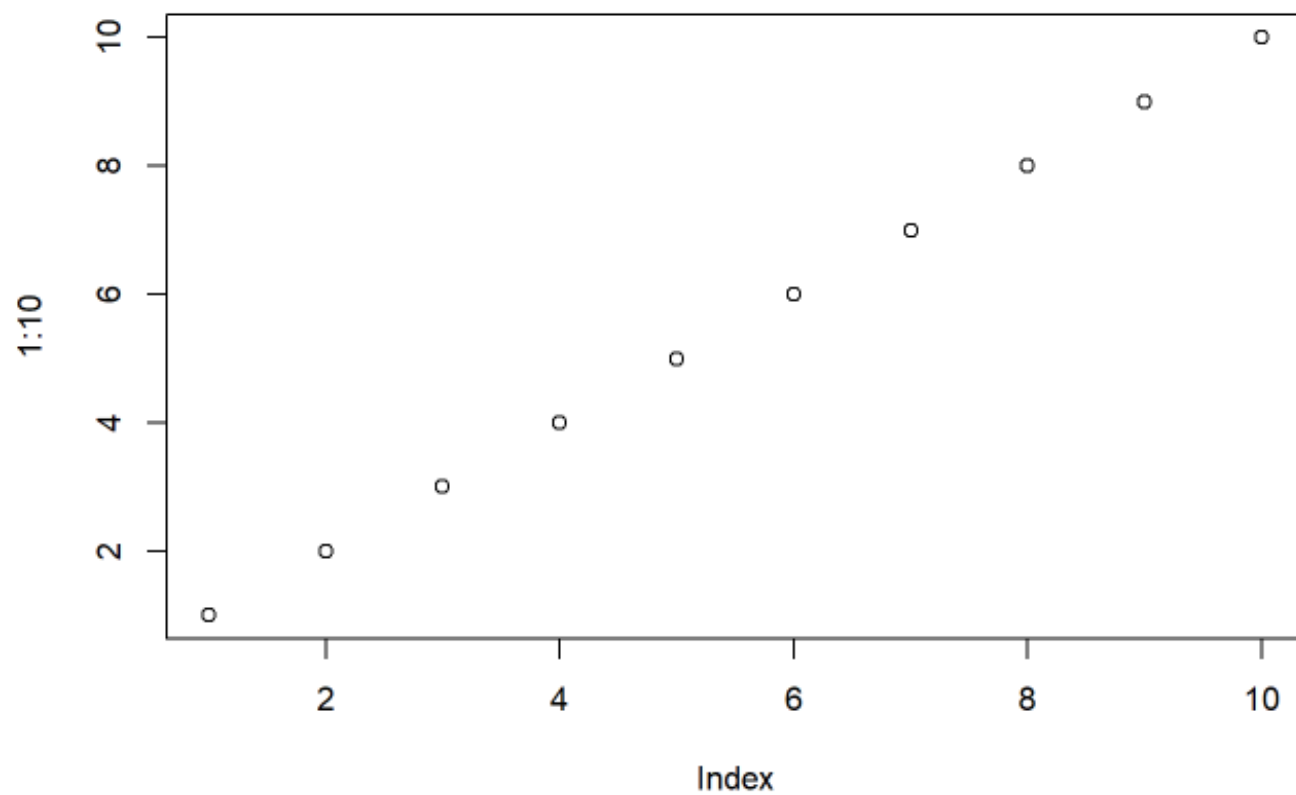
```
## [1] 5.551115e-17
```

使用指南：长篇文档

graphics

It is easy.

```
plot(1:10)
```



使用指南：长篇文档

- ▶ 更多关于Rmarkdown的知识可以参见下面的链接

<https://rmarkdown.rstudio.com/lesson-1.html>