

## 第二章：R 语言基础

---

王敏杰

2020 年 7 月 26 日

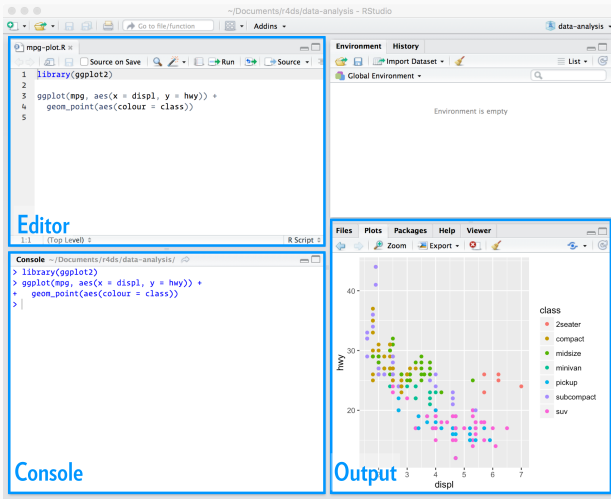
四川师范大学

# 开始

---

# 开始

安装完毕后，从 windows 开始菜单，点开 rstudio 图标，就打开了 rstudio 的窗口，界面效果如下



# RStudio 非常友好

想要运行一段 R 代码，只需要在 RStudio 控制台面板最下面 (Console) 一行内键入 R 代码，然后回车即可。比如

```
1 + 1
```

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

```
log(8)
```

```
#> [1] 2.08
```

```
1:12
```

```
#> [1] 1 2 3 4 5 6 7 8 9 10 11 12
```

# 对象

## 一切都是对象

在 R 中存储的数据称为对象，R 语言数据处理实际上就是不断的创建和操控这些对象。

## 创建对象

创建一个 R 对象，首先确定一个名称，然后使用赋值操作符 `<-`，将数据赋值给它。比如，如果想给变量 `x` 赋值为 5，在命令行中可以这样写 `x <- 5`，然后回车。

```
x <- 5
```

## 打印对象

当键入 `x` 然后回车，就打印出 `x` 的值

# 对象

## 创建对象

```
l <- "hello world"
```

## 访问对象

```
l  
#> [1] "hello world"
```

# 对象

## 创建一个序列

```
d <- 1:10
```

## 访问对象

```
d
```

```
#> [1] 1 2 3 4 5 6 7 8 9 10
```

## 数据类型

Type	Examples
numeric	0, 1, -2, 3.1415, 0.0005
character	"Amelia", "Agree", "31"
logical	TRUE, FALSE
factor	a c c b Levels: a b c



# 数据类型

- 数值型

```
3
```

```
#> [1] 3
```

```
5000
```

```
#> [1] 5000
```

```
3e+06
```

```
#> [1] 3e+06
```

```
class(0.0001)
```

```
#> [1] "numeric"
```

# 数据类型

- 字符串型

```
"hello"
```

```
#> [1] "hello"
```

```
"girl"
```

```
#> [1] "girl"
```

```
"1"      # 注意 1 和 "1" 的区别
```

```
#> [1] "1"
```

```
class("1")
```

```
#> [1] "character"
```

# 数据类型

- 逻辑型

```
TRUE
```

```
#> [1] TRUE
```

```
FALSE
```

```
#> [1] FALSE
```

```
3 < 4
```

```
#> [1] TRUE
```

```
class(T)
```

```
#> [1] "logical"
```

```
3 < 4
```

```
#> [1] TRUE
```

- 因子型

```
fac <- factor(c("a", "b", "c"))
```

```
fac
```

```
#> [1] a b c
```

```
#> Levels: a b c
```

```
class(fac)
```

```
#> [1] "factor"
```

# 数据结构

## 向量

- 用 `c` 函数将一组数据构造成为向量，要求每个元素用逗号分隔，且每个元素的数据类型是一致的

```
d <- c(2, 4, 3, 1, 5, 7)
```

```
d
```

```
#> [1] 2 4 3 1 5 7
```

```
t <- c("2", "4", "3", "1", "5", "7")
```

```
t
```

```
#> [1] "2" "4" "3" "1" "5" "7"
```

长度为 1 的向量

# 数据结构

## 矩阵

- 可以用 `matrix` 函数创建

```
m <- matrix(c(2, 4, 3, 1, 5, 7),  
            nrow = 2,  
            ncol = 3,  
            byrow = TRUE  
)
```

m

```
#>      [,1] [,2] [,3]  
#> [1,]    2    4    3  
#> [2,]    1    5    7
```

# 数据结构

## 数组

- array 函数生成 n 维数组

```
ar <- array(c(11:14, 21:24, 31:34), dim = c(2, 2, 3))
```

```
ar
```

```
#> , , 1
```

```
#>
```

```
#>      [,1] [,2]
```

```
#> [1,]    11    13
```

```
#> [2,]    12    14
```

```
#>
```

```
#> , , 2
```

```
#>
```

```
#>      [,1] [,2]
```

# 数据结构

## 列表

- 与 c 函数创建向量的方式相似，元素之间用逗号分开。不同的是，列表允许每个元素不同的数据类型（数值型，字符型，逻辑型等），而向量要求每个元素的数据类型必须相同。

```
list1 <- list(100:110, "R", c(2, 4, 3, 1, 5, 7))  
list1  
#> [[1]]  
#> [1] 100 101 102 103 104 105 106 107 108 109 110  
#>  
#> [[2]]  
#> [1] "R"  
#>
```



# 数据结构

## 数据框

- data.frame 函数构建

```
df <- data.frame(  
  name = c("ace", "bob", "carl", "kaite"),  
  age = c(21, 14, 13, 15),  
  sex = c("girl", "boy", "boy", "girl")  
)
```

df

```
#>      name age  sex  
#> 1    ace  21 girl  
#> 2    bob  14  boy  
#> 3   carl  13  boy  
#> 4  kaite  15 girl
```

# 数据结构

## 数据框

R 对象的数据结构 (向量、矩阵、数组、列表和数据框), 总结如下

Vector

"1"	"R"	"TRUE"
-----	-----	--------

character

Matrix

"1"	"R"	"TRUE"
"2"	"S"	"FALSE"
"3"	"T"	"TRUE"

character

List

1	"R"	TRUE
---	-----	------

numeric   character   logical

data frame

1	"R"	TRUE
2	"S"	FALSE
3	"T"	TRUE

numeric   character   logical

# 函数

R 语言的强大在于使用函数操控各种对象，你可以把对象看作是名词，而函数看作是动词。我们用一个简单的例子，`sum()` 来演示函数如何工作的。`sum()` 后的结果可以直接显示出来，

```
sum(5, 10)
#> [1] 15
```

也可以赋名。比如下面代码，首先计算  $5 + 10$  然后赋给新创建的对象 `y`，并在第二行中打印出来对象 `y` 的值

```
y <- sum(5, 10)
y
#> [1] 15
```

## 更多函数

除了 `sum()` 求和函数，R 语言有很多很多函数

```
mean(1:6)
```

```
#> [1] 3.5
```

```
abs(1:6)
```

```
#> [1] 1 2 3 4 5 6
```

```
round(3.14159)
```

```
#> [1] 3
```

```
x <- seq(1, 100)
```

```
sum(x)
```

```
#> [1] 5050
```

## 什么是脚本

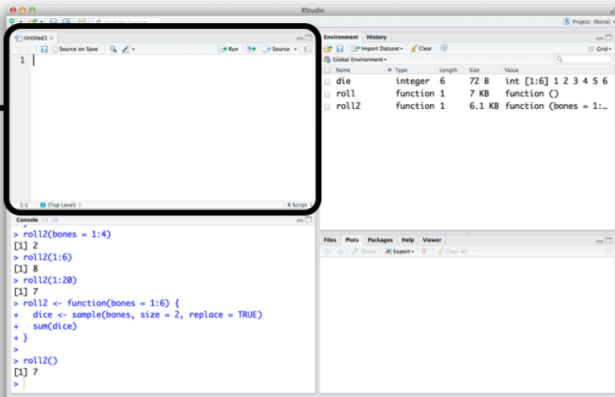
如果我们已经写好了一段 R 程序，我们可以保存为脚本文件，脚本文件通常以.R 作为文件的后缀名。比如我们可以将刚才创建 `x` 和 `y` 对象的命令，保存为脚本文件 `my_script.R`。这样我们可以在其它时间修改和重新运行它。

## 创建脚本

在 RStudio 中，你可以通过菜单栏依此点击 `File > New File > R Script` 来创建一个新的脚本。强烈建议大家在运行代码之前，使用脚本的形式编写和编辑自己的程序，养成这样的习惯后，你今后所有的工作都有案可查，并且具有可重复性。

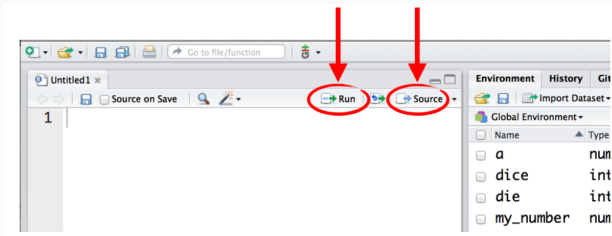
# 创建脚本

New script



# 运行脚本

- 点击 Run 运行光标所在行
- 点击 Source 运行整个脚本





# 宏包

R 语言的强大还在于各种宏包，一般在The Comprehensive R Archive Network (CRAN)下载安装。

可以用如下命令安装宏包：

```
# 安装单个包
```

```
install.packages("tidyverse")
```

```
# 安装多个包
```

```
install.packages(c("ggplot2", "devtools", "dplyr"))
```

# 如何获取帮助

- 记住和学习所有的函数几乎是不可能的
- 打开函数的帮助页面 (Rstudio 右下面板的 Help 选项卡)

```
?sqrt
```

```
?gather
```

```
?spread
```

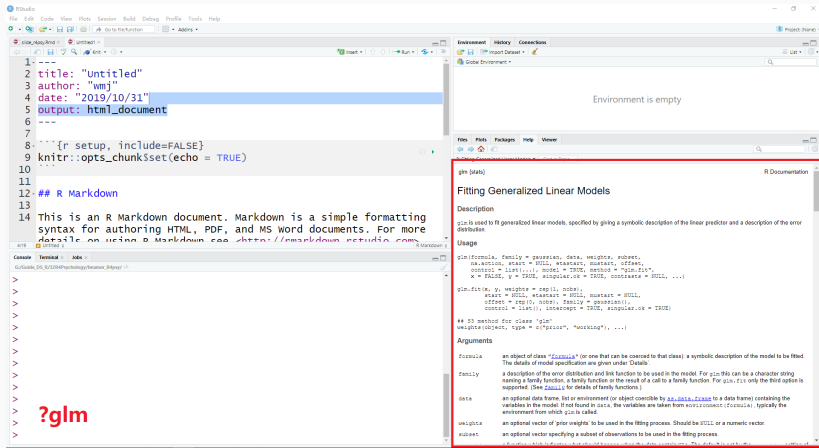
```
?ggplot2
```

```
?scale
```

```
?map_dfr
```

# 如何获取帮助

## 快速获取帮助，是 R 的又一个优良特性



The screenshot shows the RStudio interface. The left pane contains an R Markdown document with the following code:

```
1. ---
2. title: "untitled"
3. author: "wmj"
4. date: "2019/10/31"
5. output: html_document
6. ---
7.
8. {r setup, include=FALSE}
9. knitr::opts_chunk$set(echo = TRUE)
10.
11.
12. ## R Markdown
13.
14. This is an R Markdown document. Markdown is a simple formatting
    syntax for authoring HTML, PDF, and MS word documents. For more
    details on using R Markdown see http://rmarkdown.rstudio.com
```

The right pane shows the "Environment" tab, which is empty. Below it, the "Files" tab is open, showing the "R Documentation" window for the `glm` function. The documentation includes the following sections:

- Description**  
`glm` is used to fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution.
- Usage**  

```
glm(formula, family = gaussian, data, weights, subset,
    na.action, start = NULL, weights, mustart, offset,
    control = list(...), model = TRUE, method = "glm.fit",
    x = FALSE, y = TRUE, singular.ok = TRUE, contrasts = NULL, ...)
glm.fit(x, y, weights = rep(1, nobs),
    start = NULL, mustart = NULL, mustart = NULL,
    offset = rep(0, nobs), family = gaussian(),
    control = list(), intercept = TRUE, singular.ok = TRUE)
## S3 method for class 'glm'
weights(object, type = c("prior", "working"), ...)
```

- Arguments**
  - formula**: an object of class `"formula"` (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under Details.
  - family**: a description of the error distribution and link function to be used in the model. For `glm`, this can be a character string naming a family function, a family function or the result of a call to a family function. For `glm`, only the third option is supported. (See `family` for details of family functions.)
  - data**: an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `model$context` (formula), typically the environment from which `glm` is called.
  - weights**: an optional vector of prior weights to be used in the fitting process. Should be `NULL` or a numeric vector.
  - subset**: an optional vector specifying a subset of observations to be used in the fitting process.

?glm