

统计学导论：R语言实验02

基础运算和变量类型

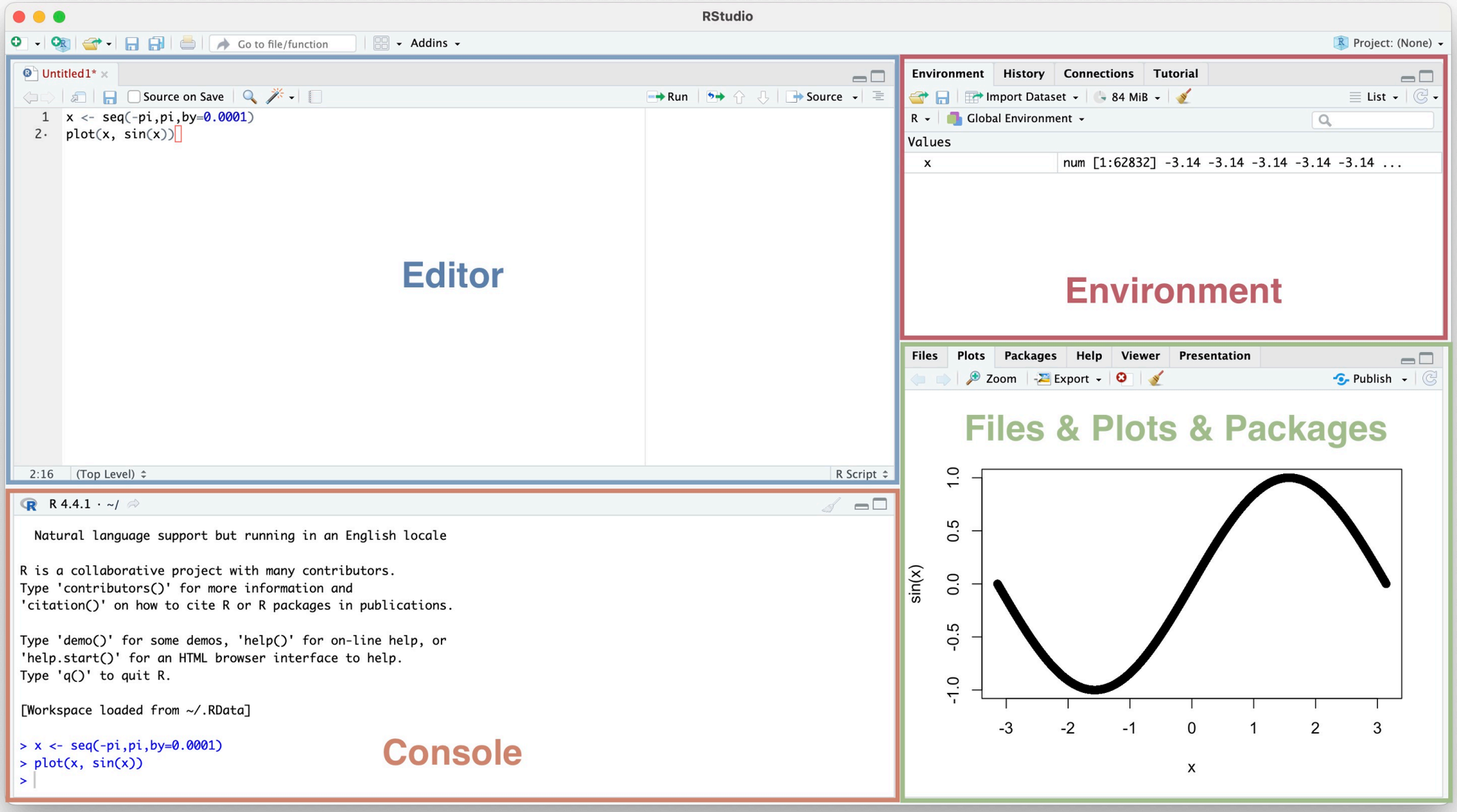
主讲人：郑盼盼

Outline

1. RStudio基础操作
2. R语言变量类型
3. R的重要数据结构
4. 基础运算和函数

01 RStudio 基础操作

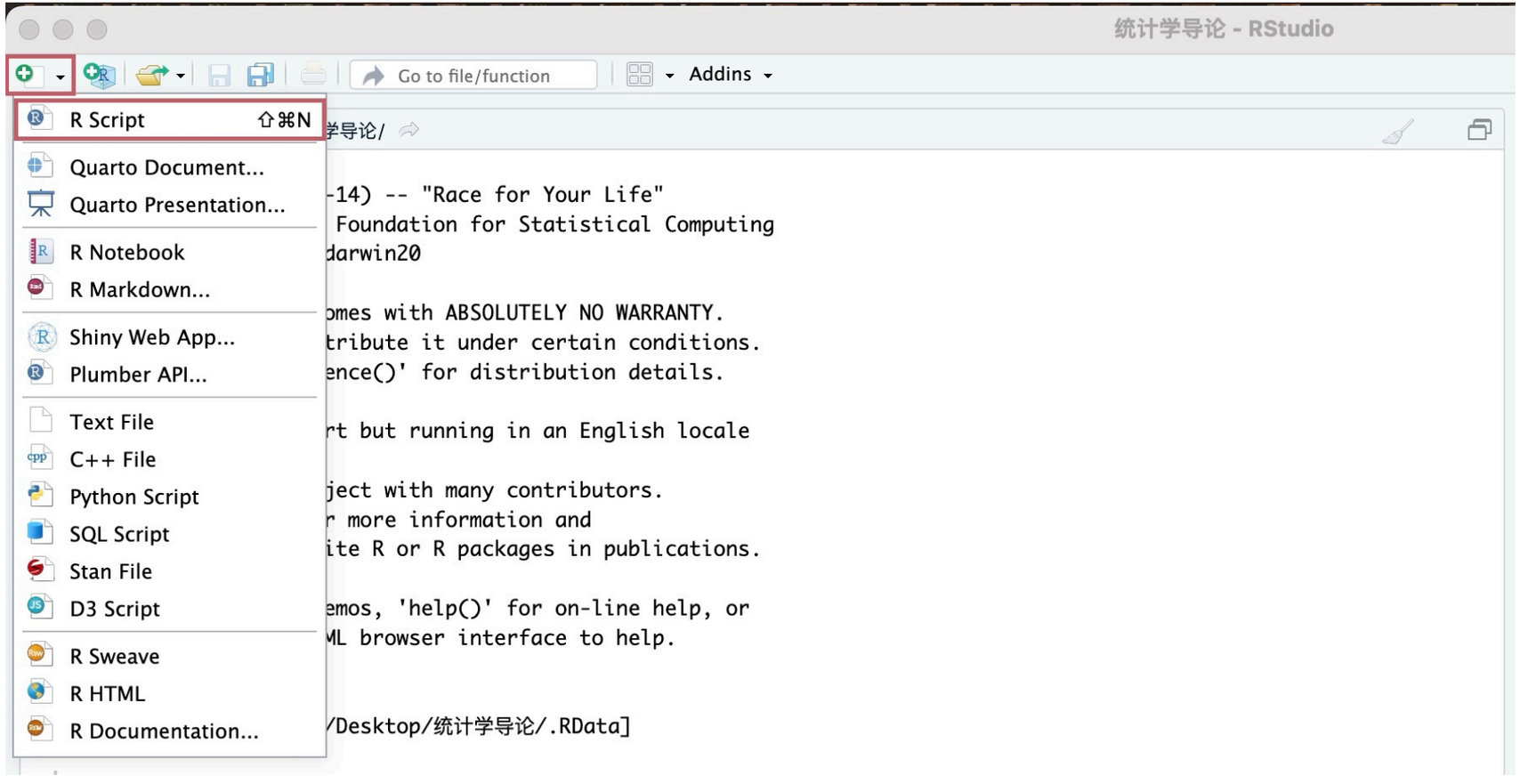
1.0 RStudio 整体界面



1.1 Editor

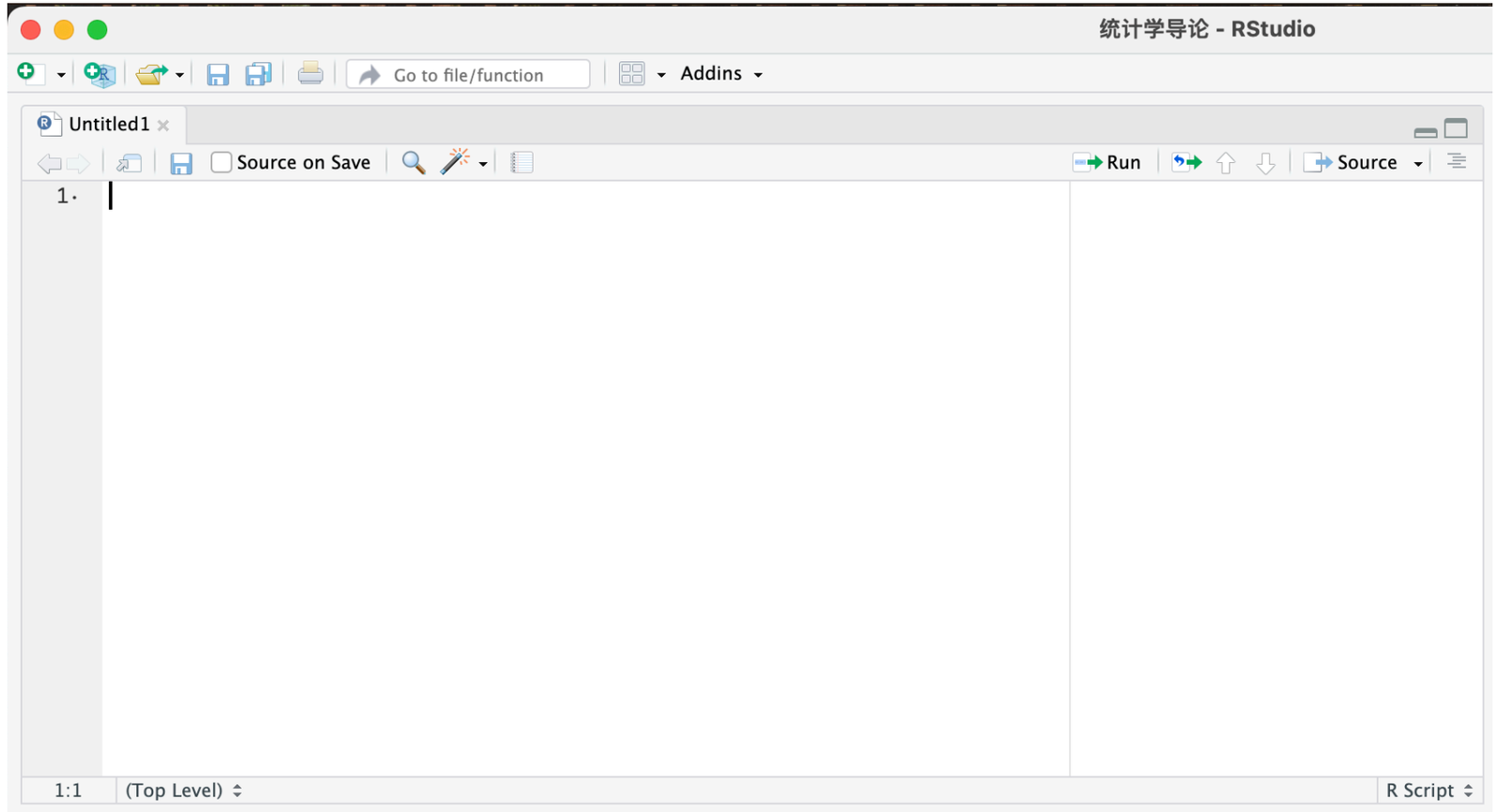
- 新建R语言脚本

点击此处!



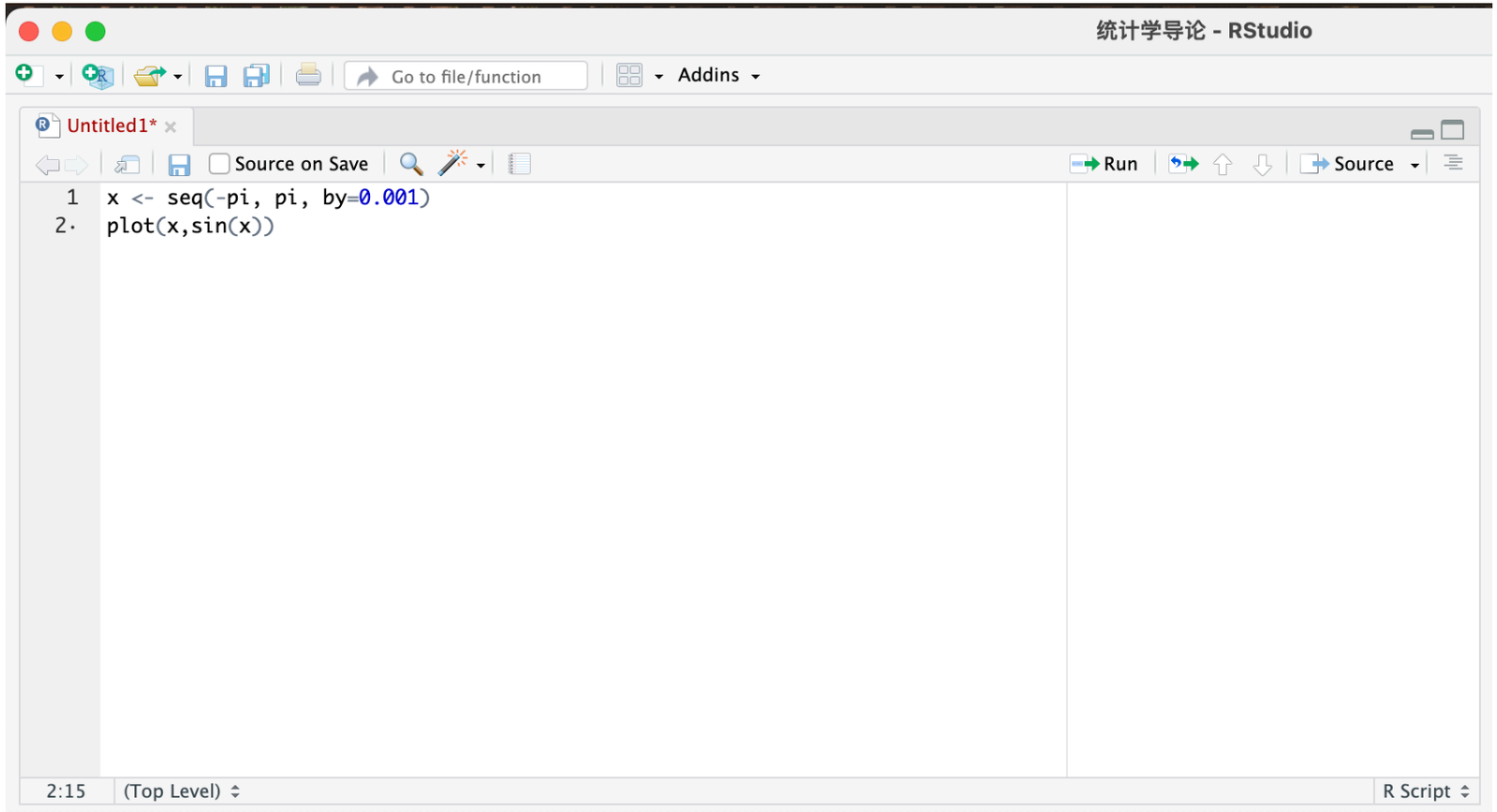
1.1 Editor

- **新建**R语言脚本: 点击后即可完成R语言脚本的新建
  - 我们可以在里面写几行简单的R语言代码



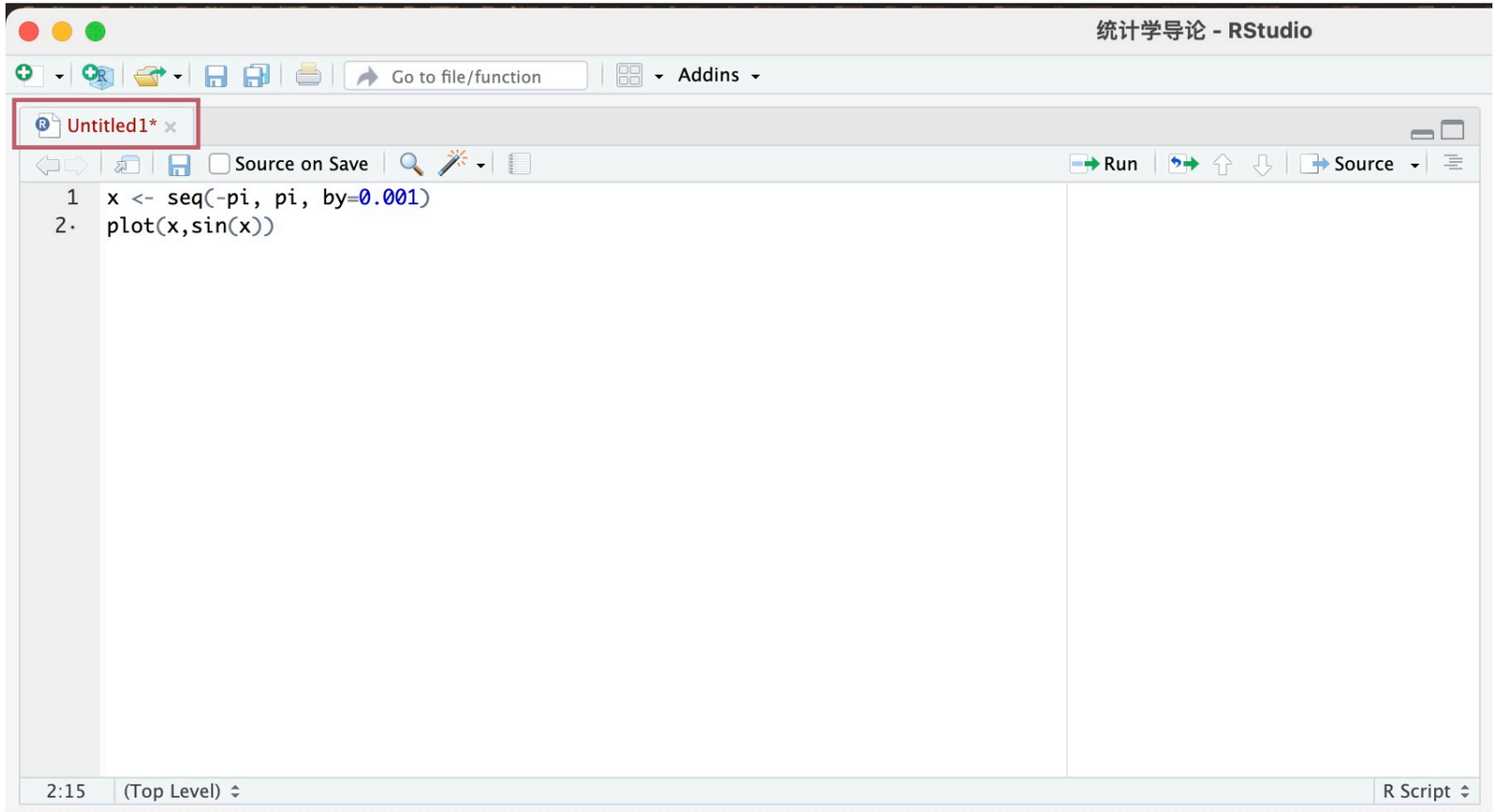
1.1 Editor

- **新建**R语言脚本: 点击后即可完成R语言脚本的新建
  - 我们可以在里面写几行简单的R语言代码



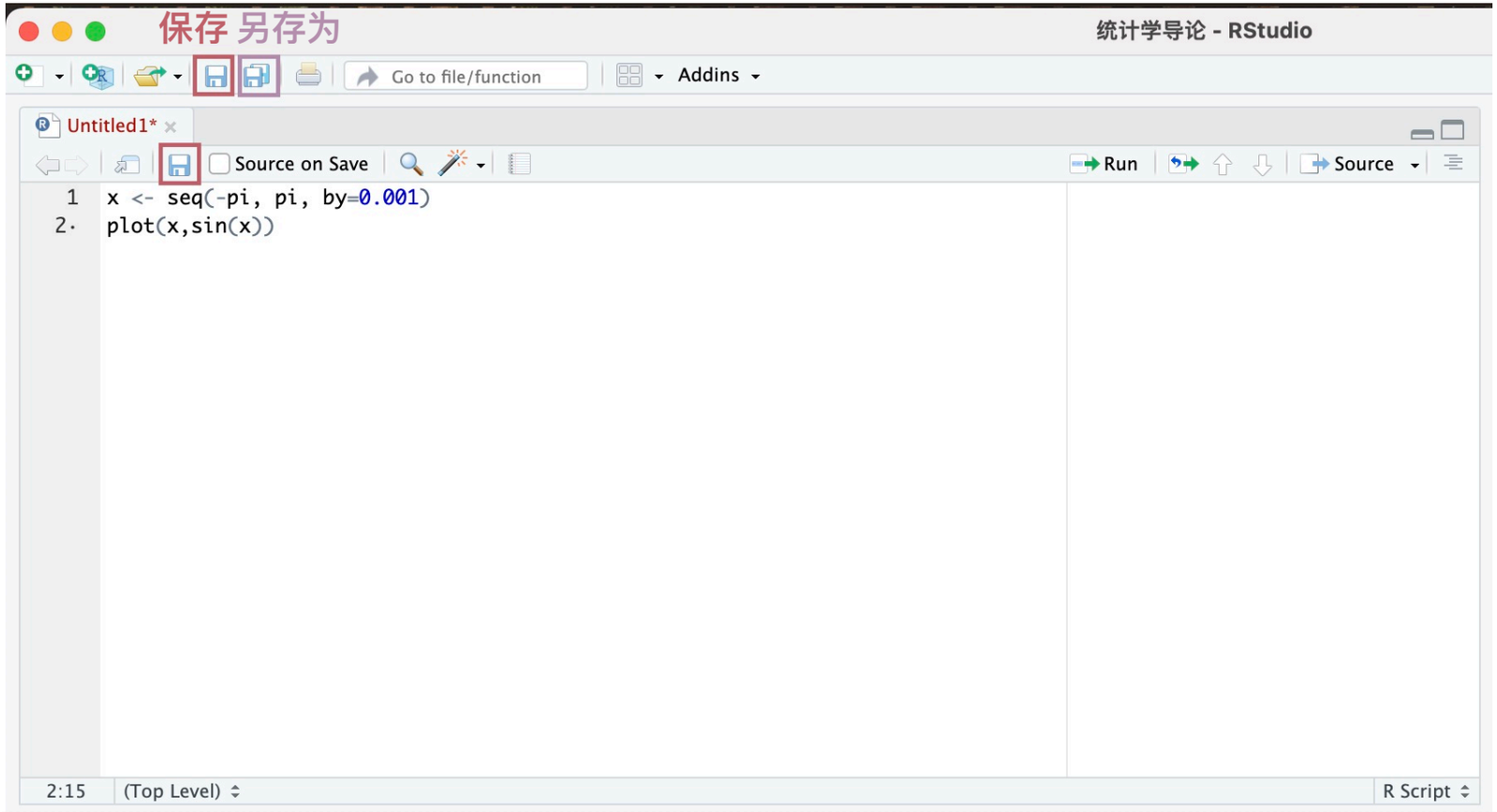
1.1 Editor

- **新建**R语言脚本: 点击后即可完成R语言脚本的新建
  - 可以发现，此时文件名变红了，旁边有个*号，这说明我们此时对于文件的更改并未得到保存！



1.1 Editor

- **新建**R语言脚本: 点击后即可完成R语言脚本的新建
- R语言脚本的**存盘**



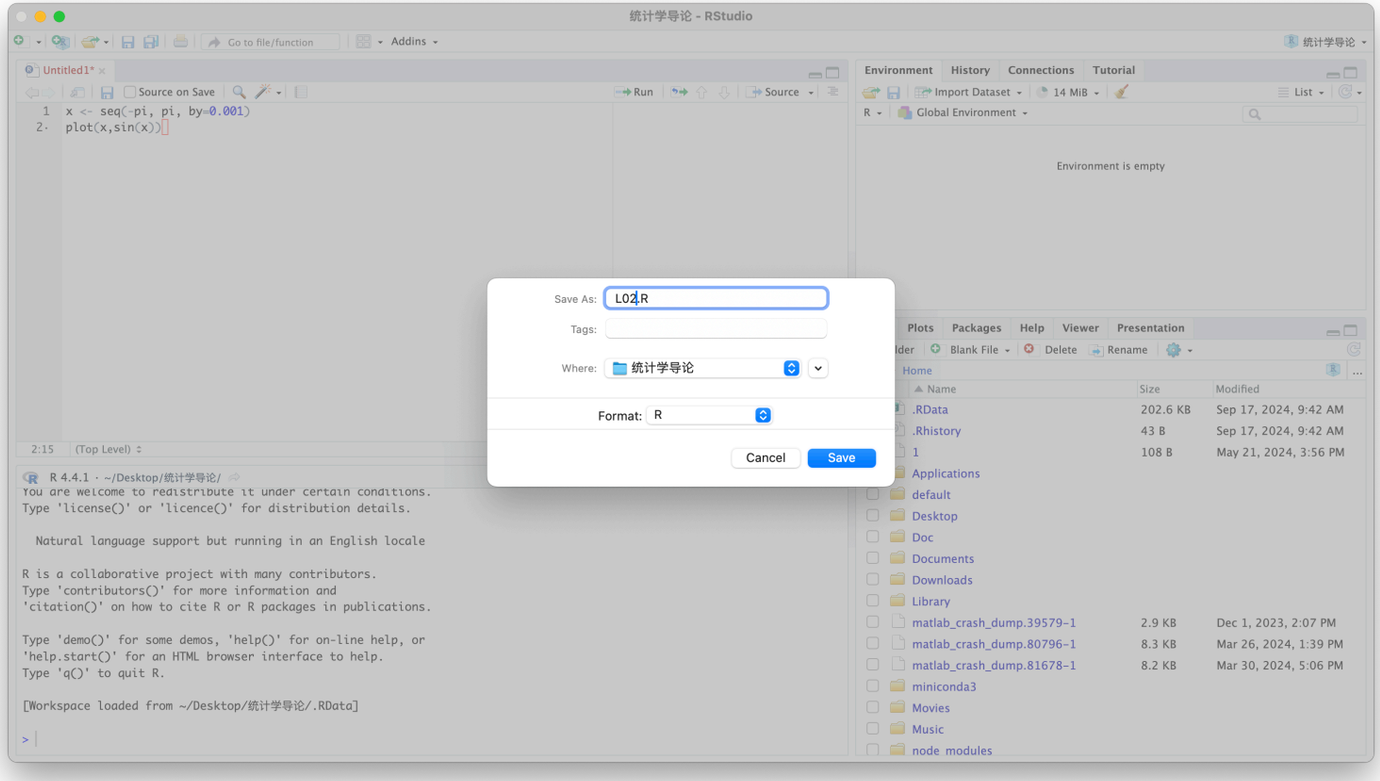
1.1 Editor

- **新建**R语言脚本: 点击后即可完成R语言脚本的新建
- R语言脚本的**存盘**
  - 第一次存储时，相对比较麻烦，需要我们指定存储的路径和文件名

tips

为了避免辛苦付诸东流，请养成及时保存修改的好习惯!

保存快捷键 `Ctrl/Command + s`



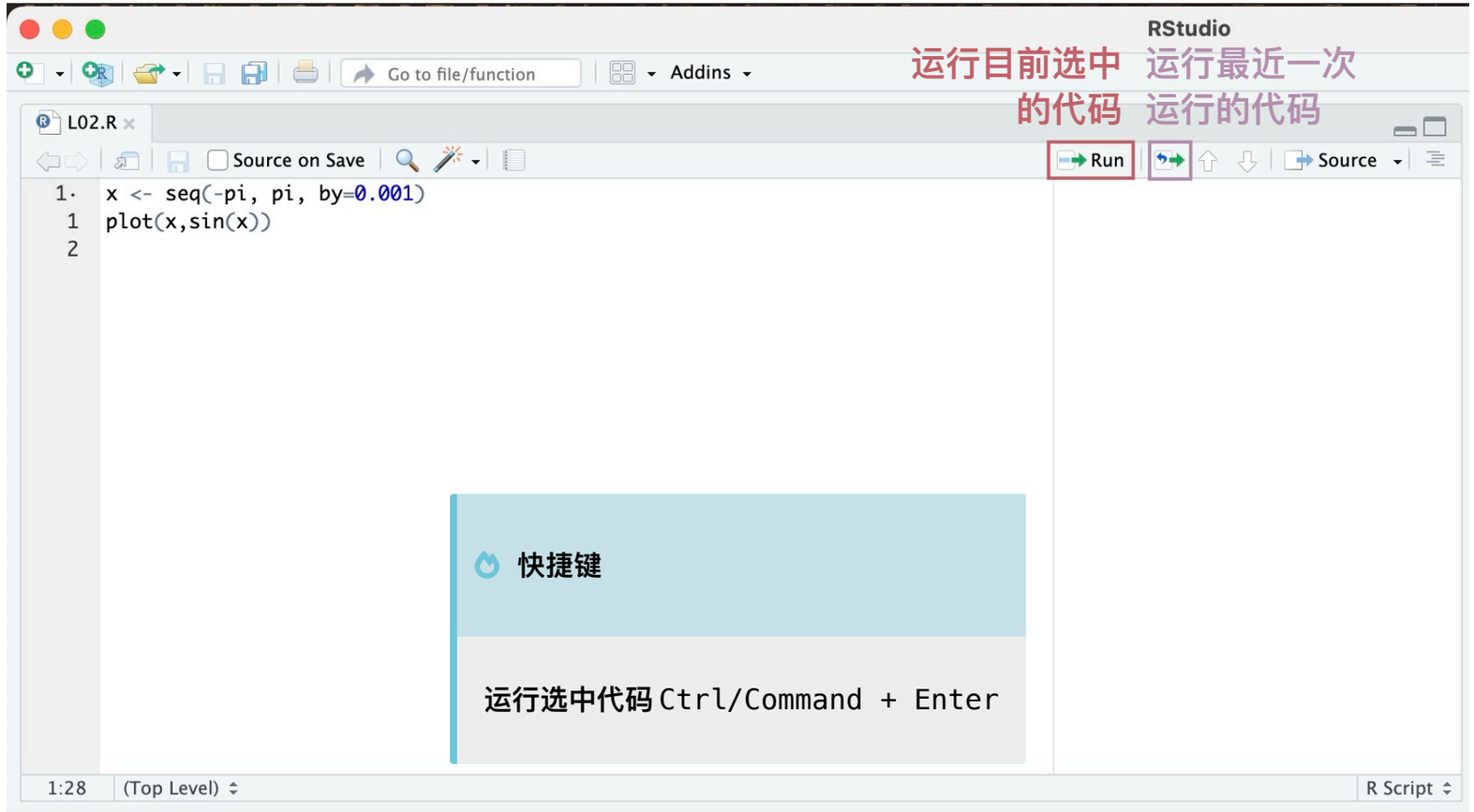
1.1 Editor

- **新建**R语言脚本: 点击后即可完成R语言脚本的新建
- R语言脚本的**存盘**
- R语言代码的**运行**



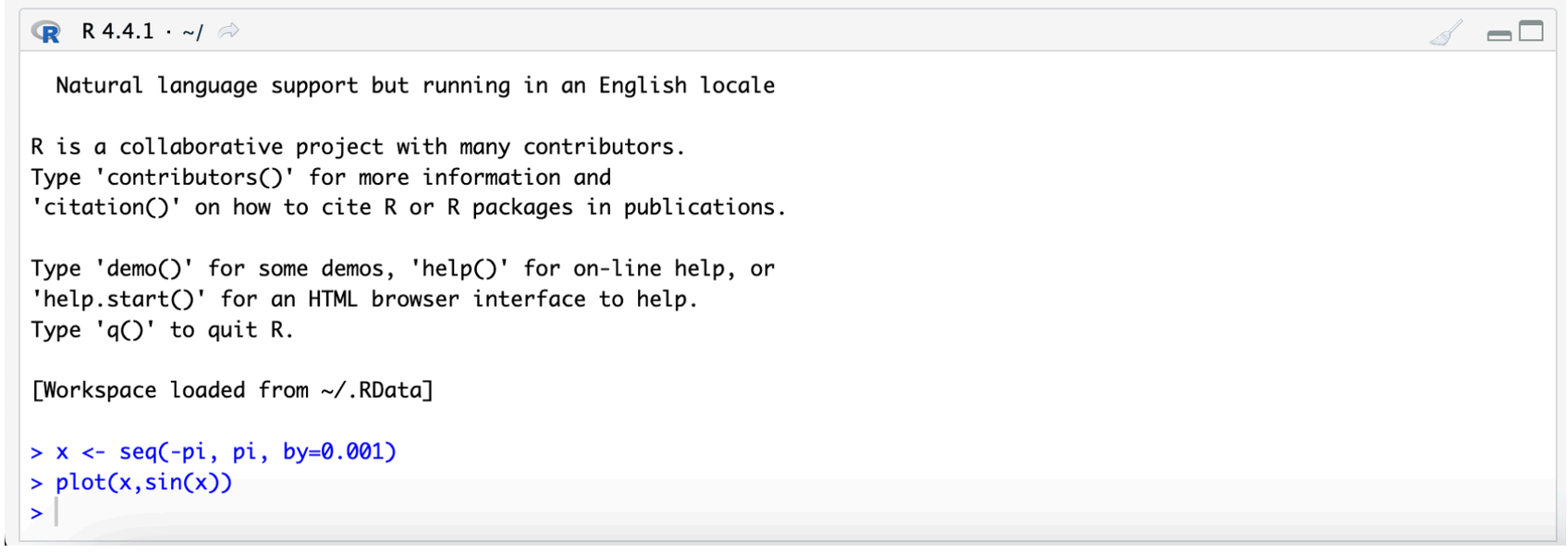
1.1 Editor

- **新建**R语言脚本: 点击后即可完成R语言脚本的新建
- R语言脚本的**存盘**
- R语言代码的**运行**



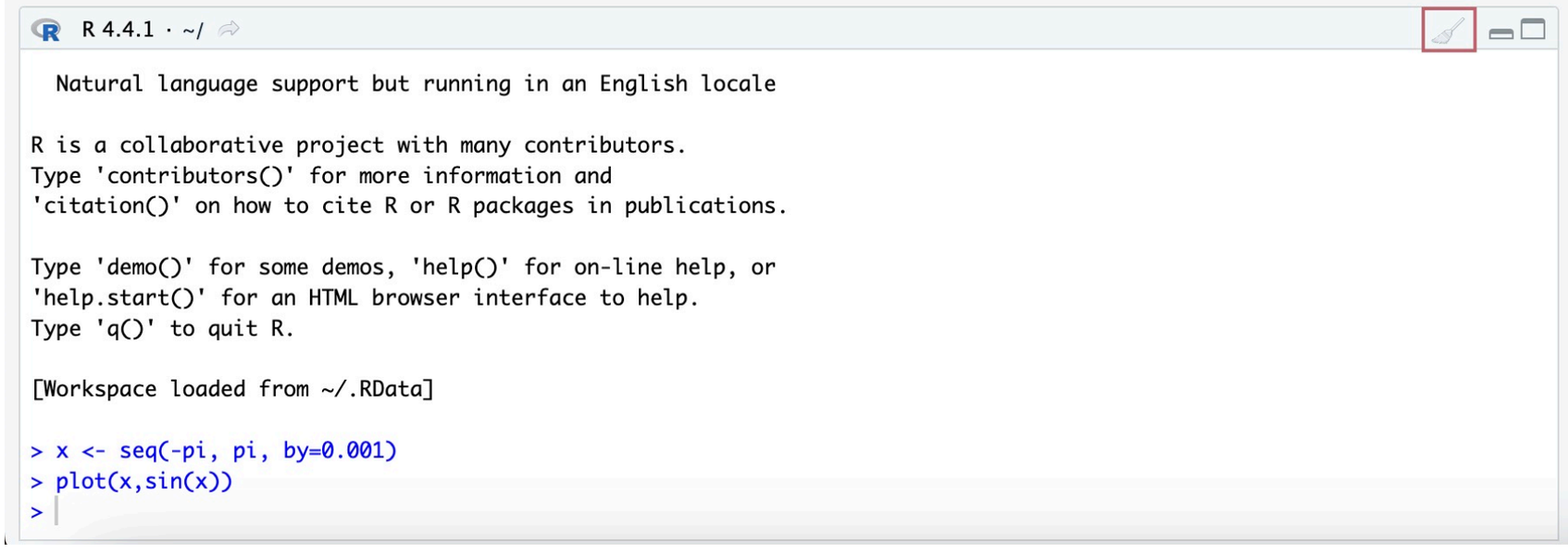
1.2 Console

- 我们可以直接在 Console 里面写R语言代码，写完后按 Enter 即可执行；
  - 当我们在 Editor 中运行代码时，RStudio会自动将运行的代码输入到下面的Console并执行。



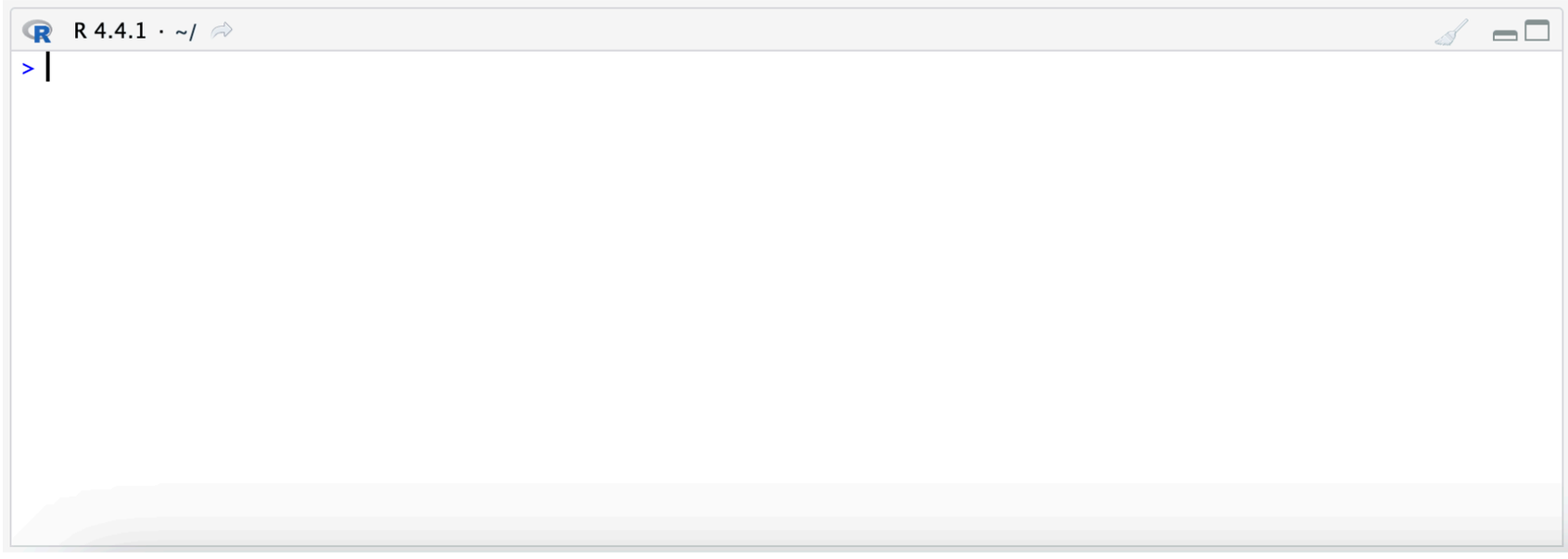
1.2 Console

- 我们可以直接在 Console 里面写R语言代码，写完后按 Enter 即可执行；
  - 当我们在 Editor 中运行代码时，RStudio会自动将运行的代码输入到下面的Console并执行。
  - 如果觉得 Console 里面的记录太多，可以点击这个小扫把清除Console里的记录（但并不会清除我们的数据）



1.2 Console

- 我们可以直接在 Console 里面写R语言代码，写完后按 Enter 即可执行；
  - 当我们在 Editor 中运行代码时，RStudio会自动将运行的代码输入到下面的Console并执行。
  - 如果觉得 Console 里面的记录太多，可以点击这个小扫把清除Console里的记录（但并不会清除我们的数据）



1.3 Environments

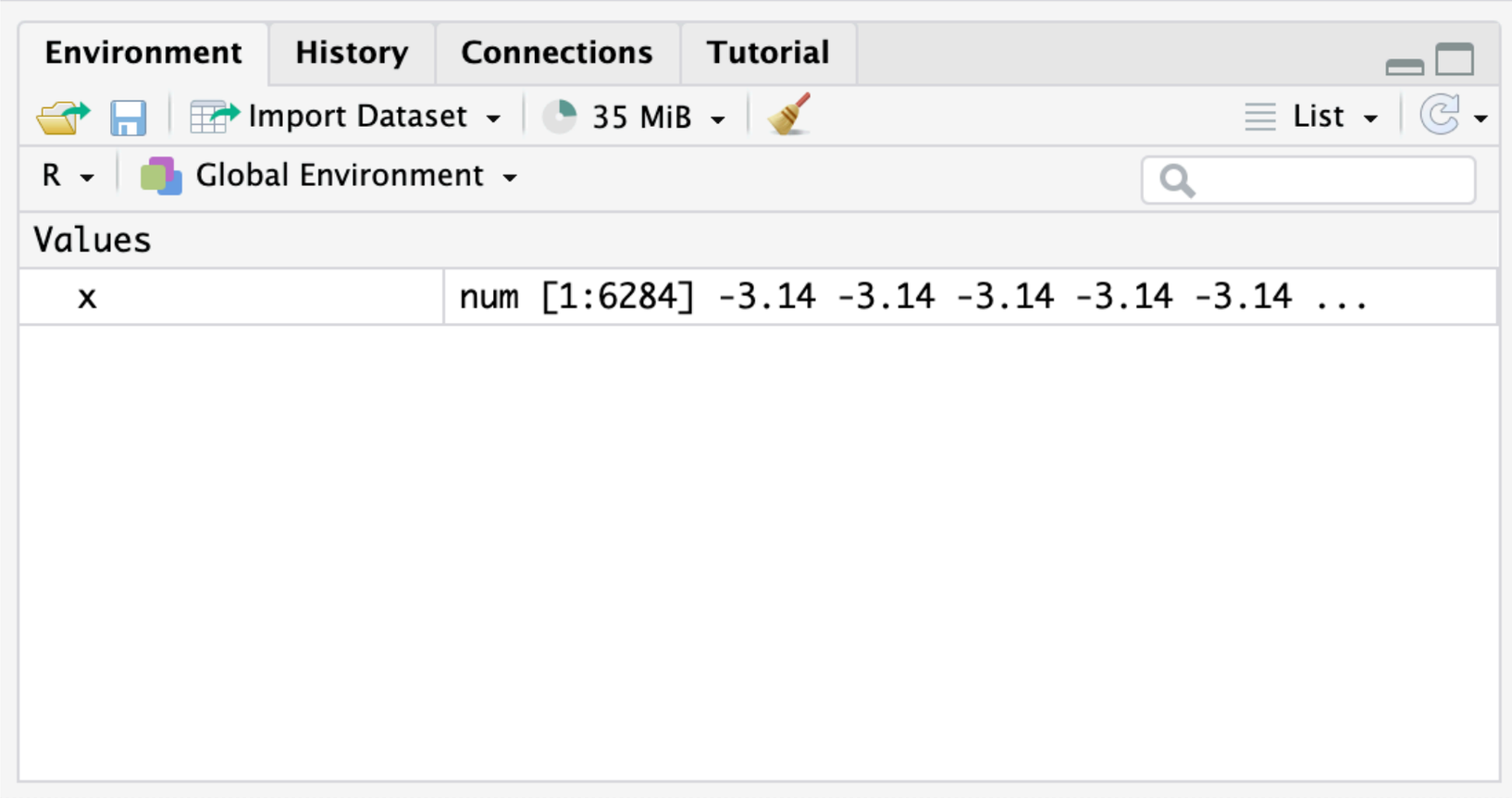
- 存储程序运行所产生的数据（变量）
  - 例如：

```
x <- seq(-pi, pi, by=0.001)
plot(x, sin(x))
```

1.3 Environments

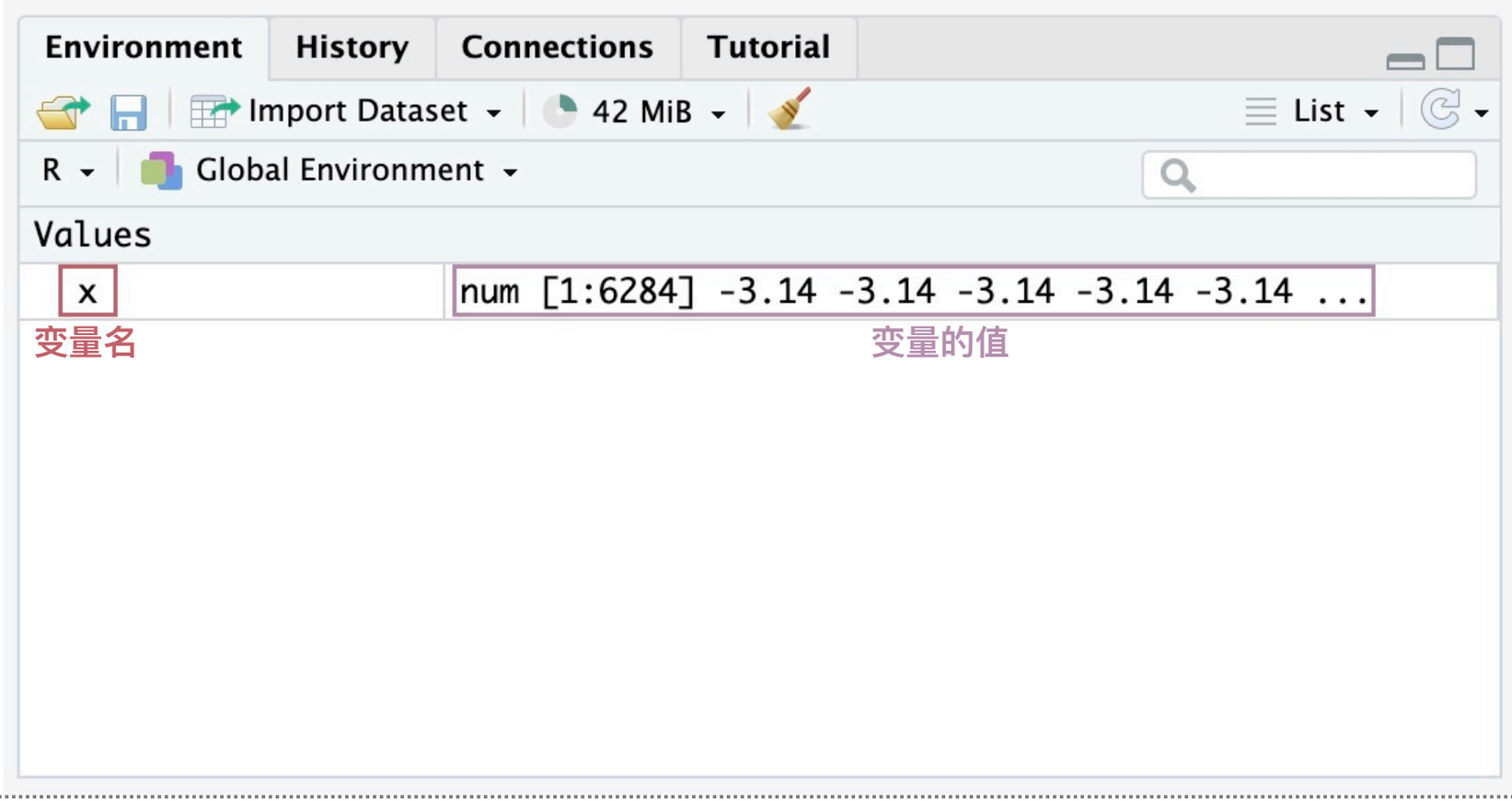
- 存储程序运行所产生的数据（变量）
  - 在Environment中能看到该变量及其数值

```
x <- seq(-pi, pi, by=0.001)
plot(x, sin(x))
```



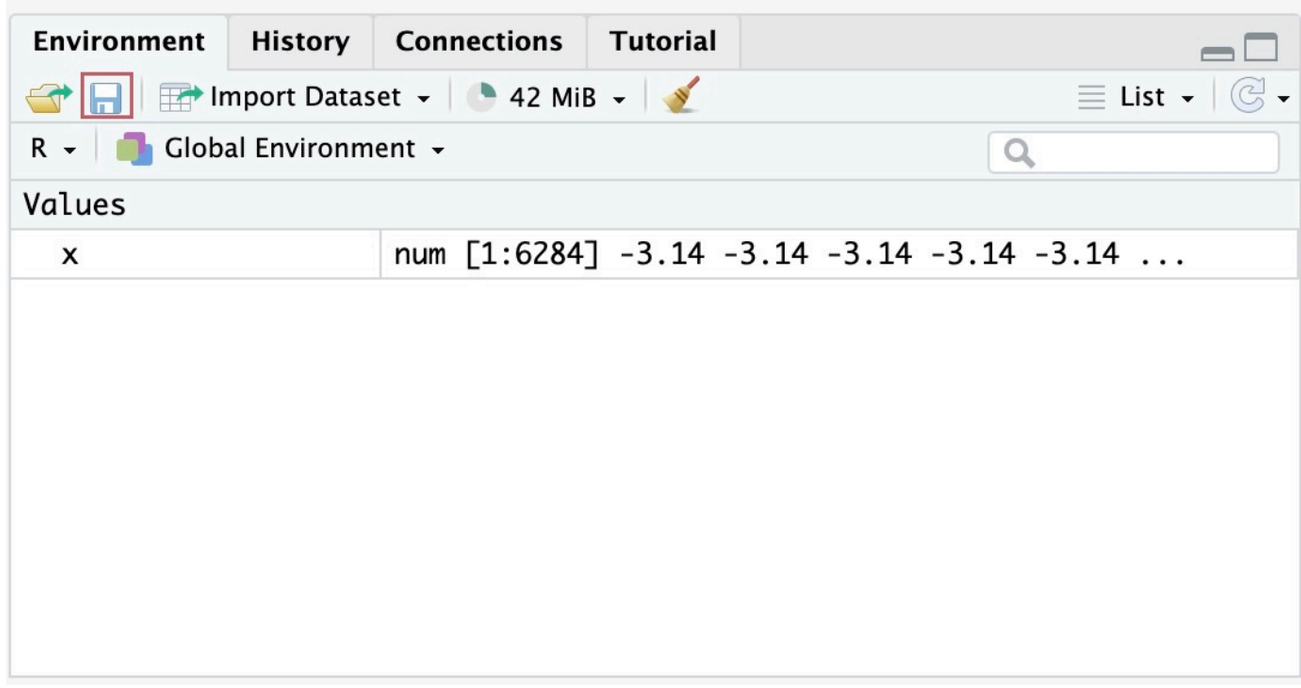
1.3 Environments

- 存储程序运行所产生的数据（变量）
  - 在Environment中能看到该变量及其数值



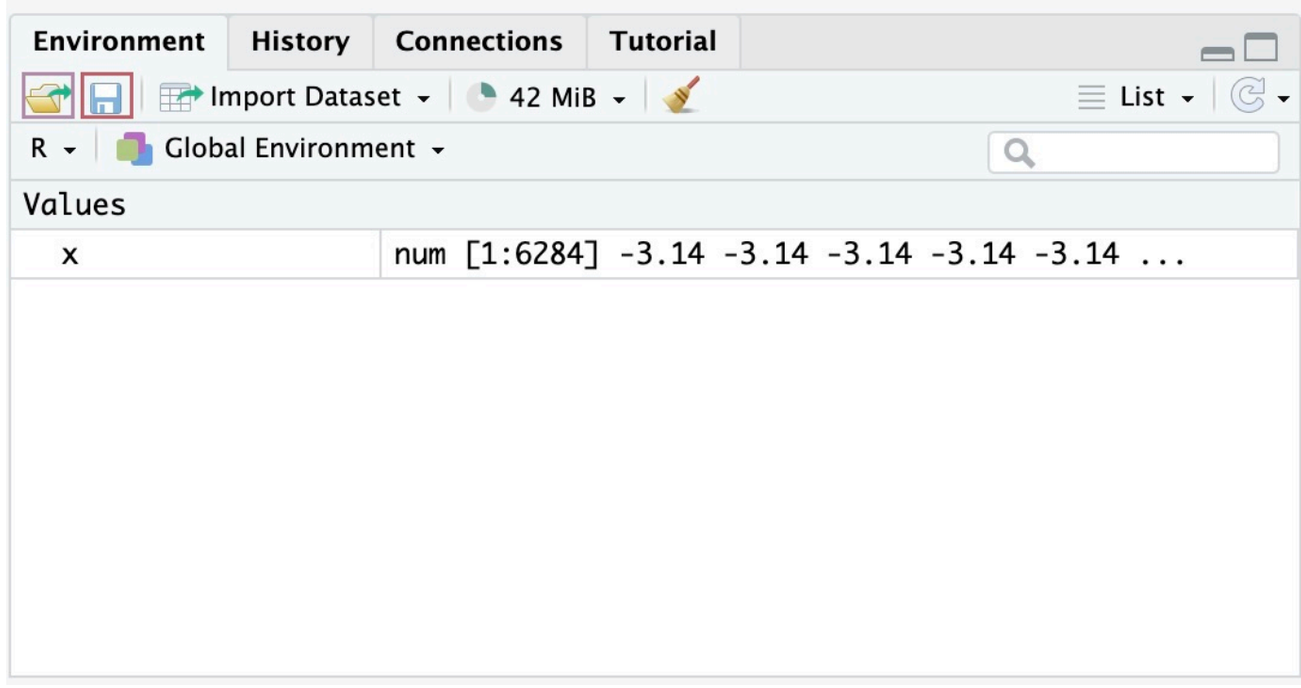
1.3 Environments

- 存储程序运行所产生的数据（变量）
  - 点击存盘按钮，可存储产生的数据（存储格式为 .RData）



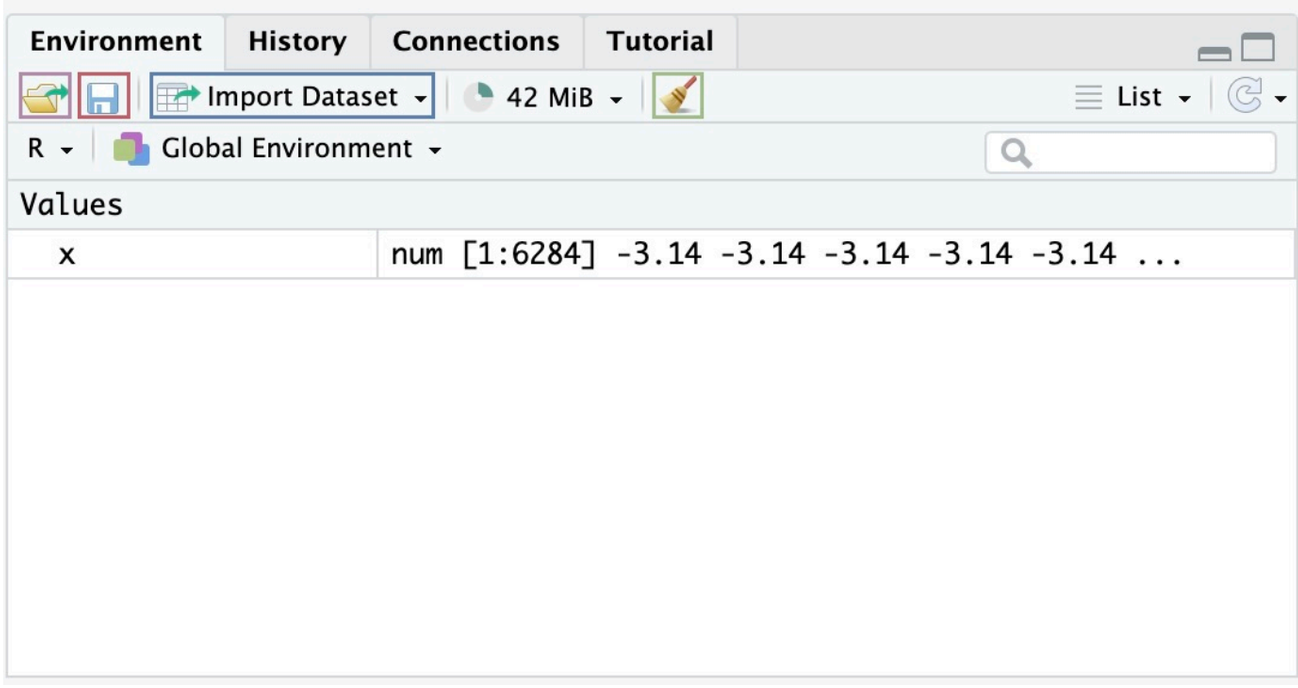
1.3 Environments

- 存储程序运行所产生的数据（变量）
  - 点击存盘按钮，可存储产生的数据（存储格式为 .RData）
  - 点击打开按钮，可以导入我们之前存储的 .RData 数据



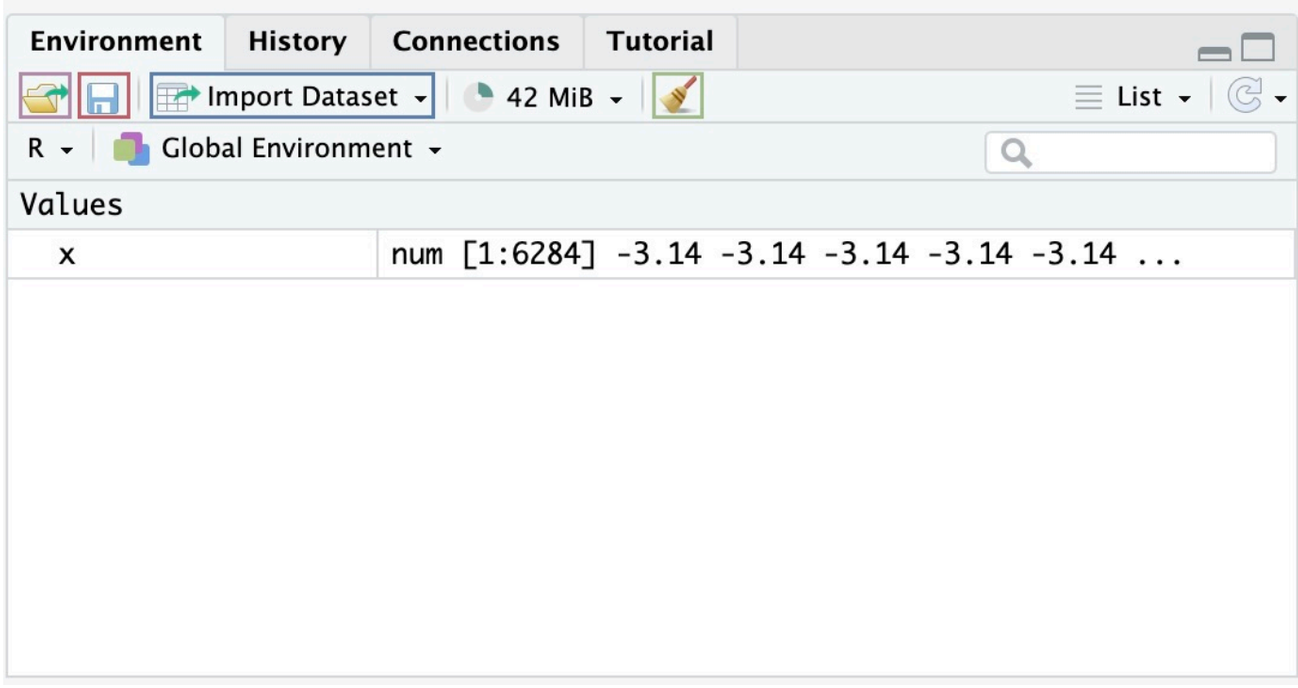
1.3 Environments

- 存储程序运行所产生的数据（变量）
  - 点击存盘按钮，可存储产生的数据（存储格式为 .RData）
  - 点击打开按钮，可以导入我们之前存储的 .RData 数据
  - 点击 Import Dataset 可导入其他格式的数据



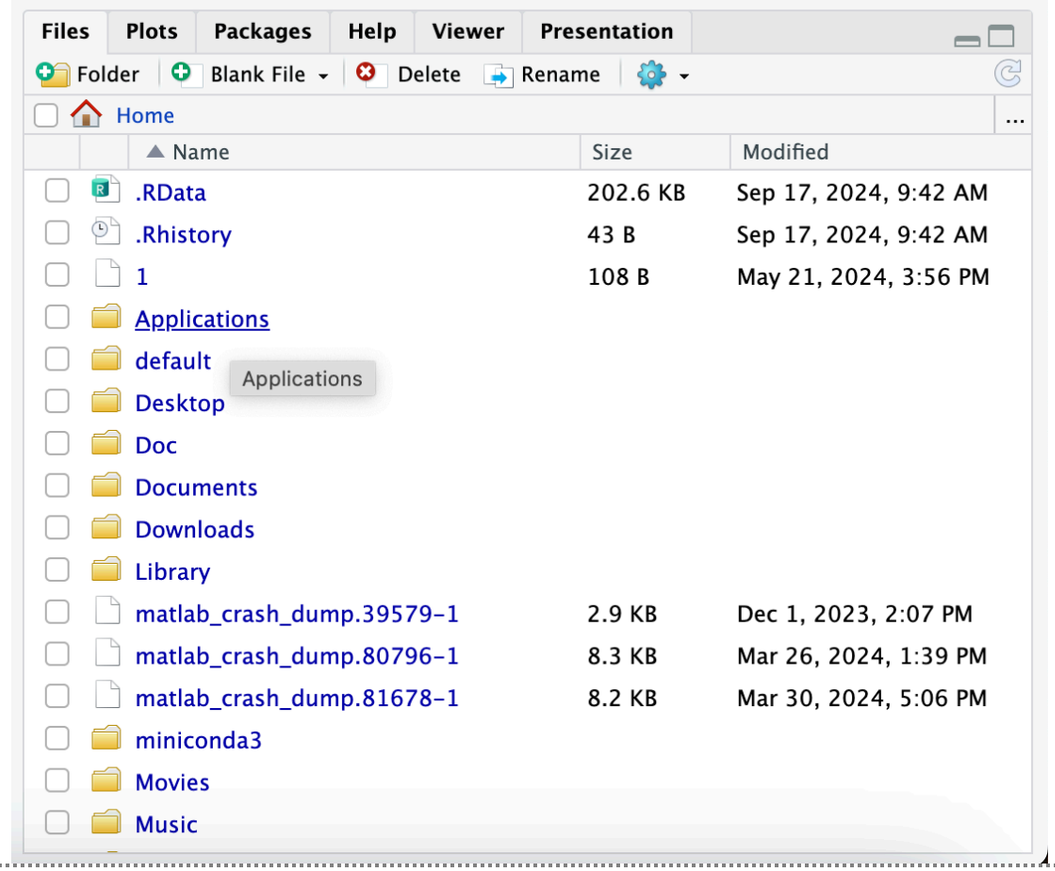
1.3 Environments

- 存储程序运行所产生的数据（变量）
  - 点击存盘按钮，可存储产生的数据（存储格式为 .RData）
  - 点击打开按钮，可以导入我们之前存储的 .RData 数据
  - 点击 Import Dataset 可导入其他格式的数据
  - 点击扫把按钮，可以清除所有的数据（谨慎使用）



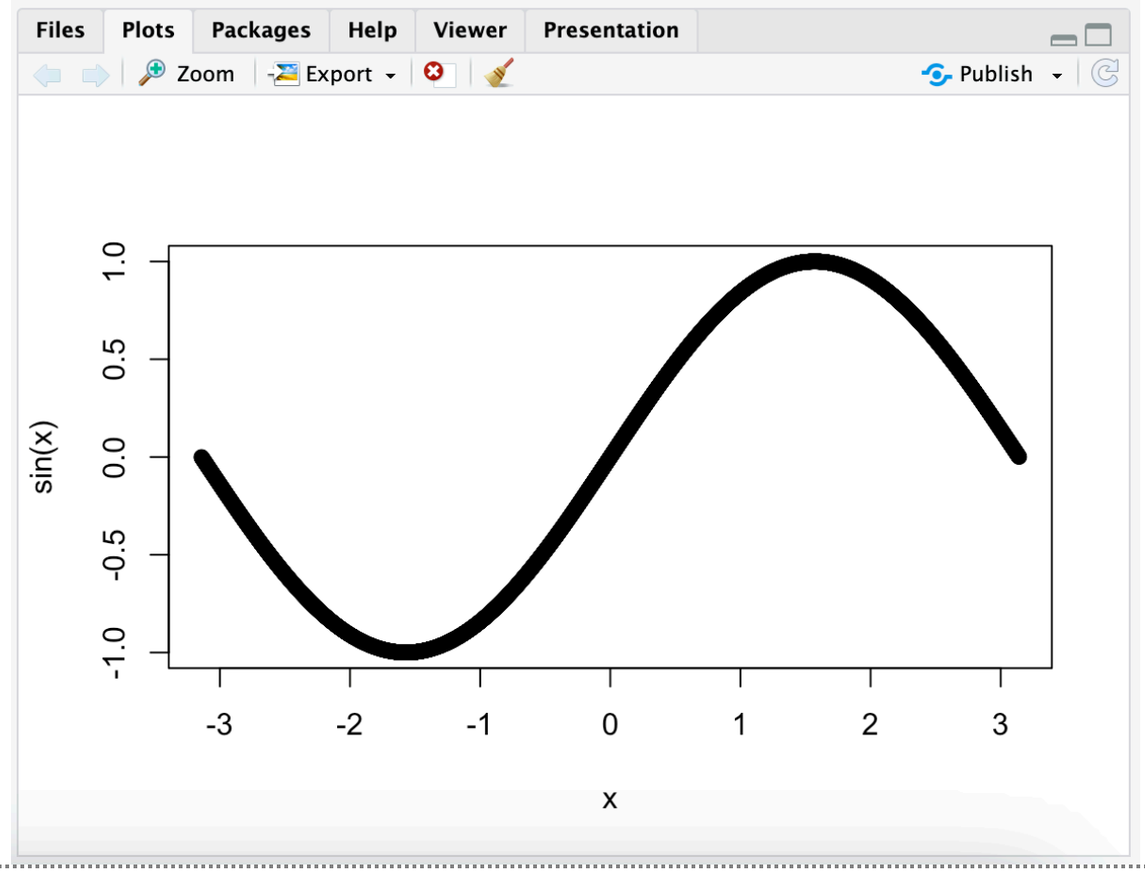
1.4 Files & Plots & Packages

- Files: 可直接打开目录中的文件，并且可以新建，删除，重命名文件和文件夹



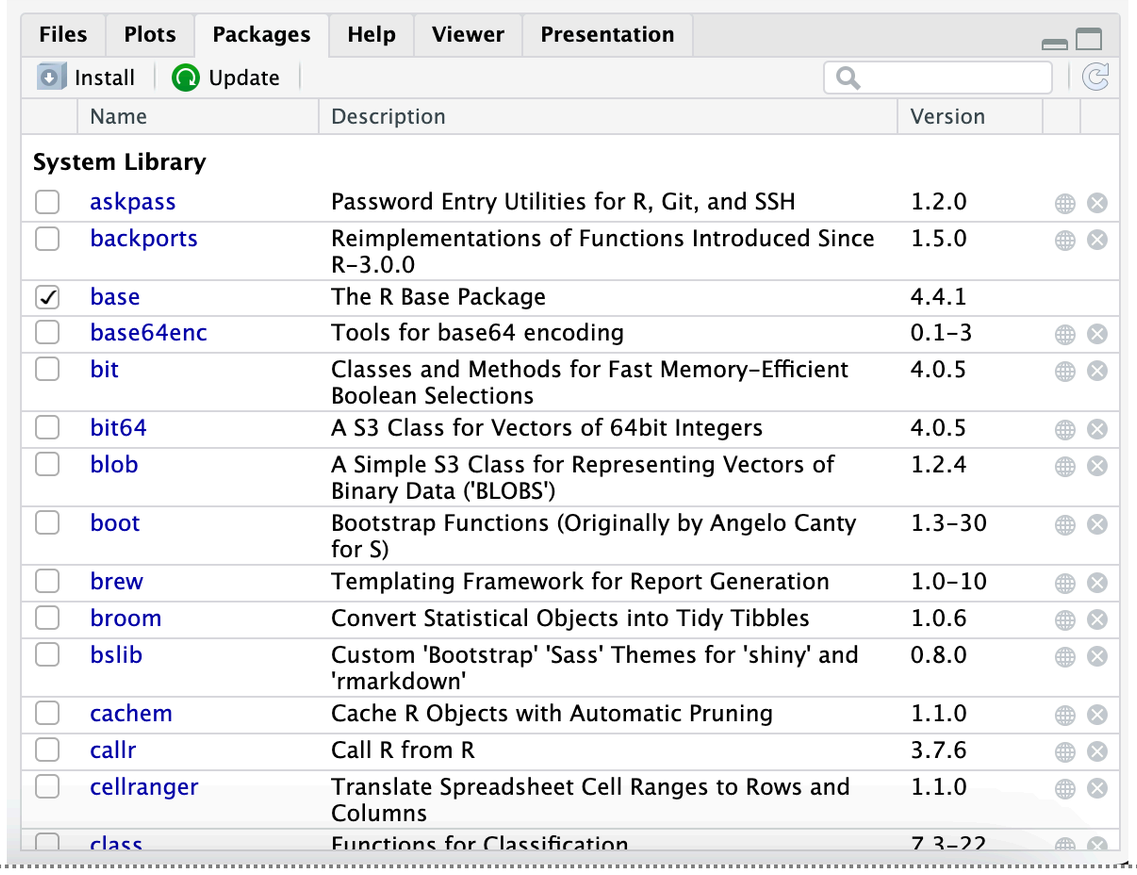
1.4 Files & Plots & Packages

- Plots: 用于显示绘图的结果，可以放大，导出图片



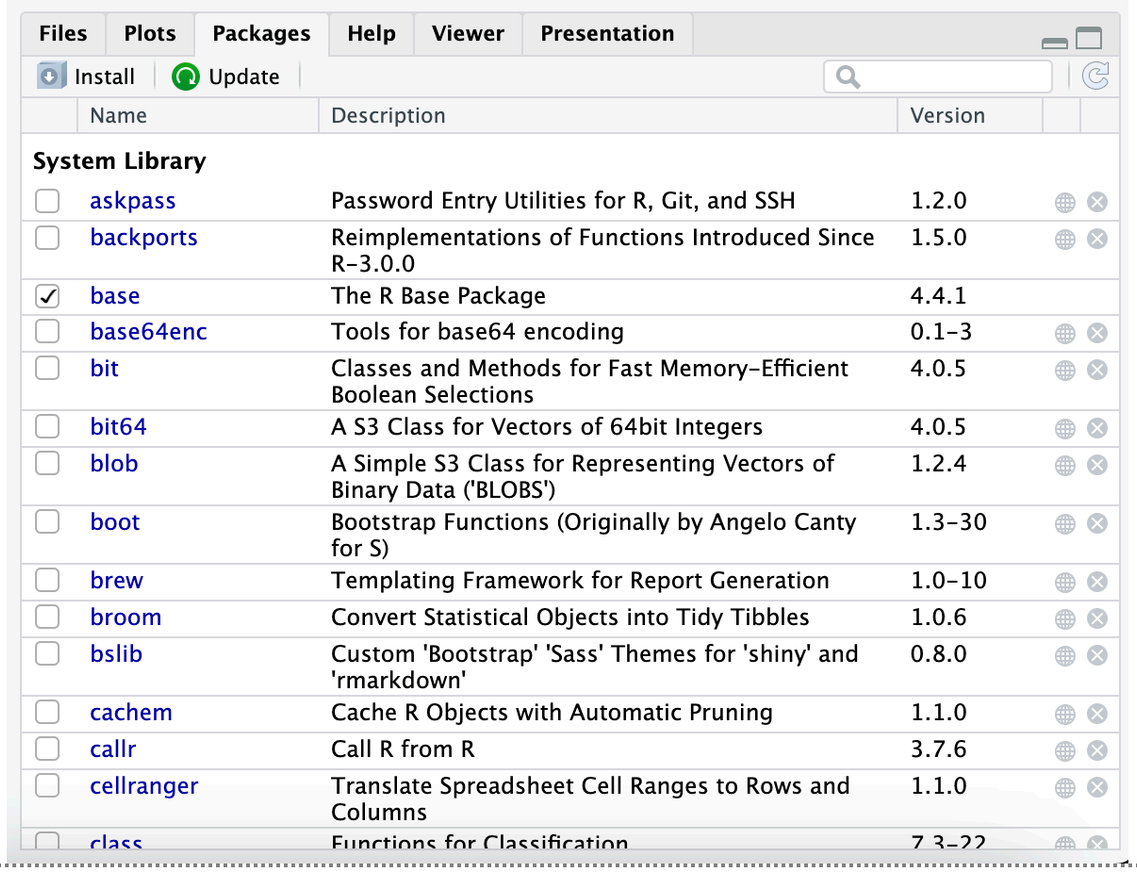
1.4 Files & Plots & Packages

- Packages: 用于安装，更新或卸载第三方包



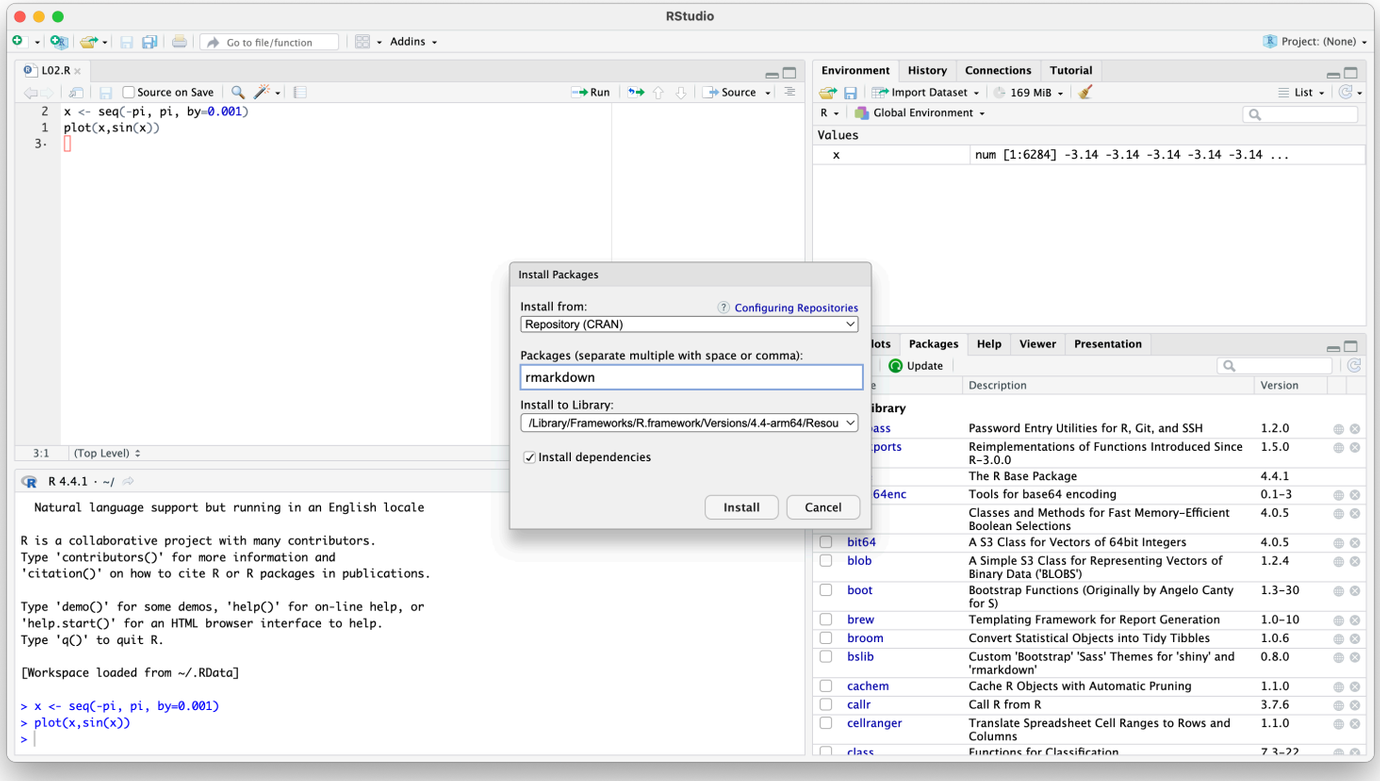
1.4 Files & Plots & Packages

- Packages: 以 `rmarkdown` 的安装为例
  - 点击 `Install`



1.4 Files & Plots & Packages

- Packages: 以 `rmarkdown` 的安装为例
  - 点击 `Install`
  - 输入 `rmarkdown` 后点击 `Install` 即可完成安装



02 R语言变量类型

2.0 变量赋值

- 我们可以使用如下的符号实现变量的赋值
  - `<-` （可以按 `Alt/Option + -` 在RStudio中快速插入
  - `=`
  - `->`
- 例如：我们希望给变量 a 赋值为 1，如下三种赋值方式是等价的

```
1 a <- 1
2 a = 1
3 1 -> a
```

2.0 变量赋值

- 我们可以使用如下的符号实现变量的赋值
  - `<-` （可以按 `Alt/Option + -` 在RStudio中快速插入
  - `=`
  - `->`
- 例如：我们希望给变量 a 赋值为 1，如下三种赋值方式是等价的

```
1 a <- 1
2 a = 1
3 1 -> a
```

> ⚠ **注意**
>
> 变量名可以包含任何大小写字母以及数字 0~9（但数字不能用于变量名开头）；且要注意区分大小写 $\mathrm{a} \neq \mathrm{A}$

2.1 变量类型

- 常见的变量类型有三种：分别为**逻辑 (logical) 变量**、**数值 (numeric) 变量**和**字符 (character) 变量**
  - 可以使用 `class(变量名)` 查看变量的类型

```r
a <- 1
class(a)

b <- TRUE
class(b)

c <- "TRUE"
class(c)
```

2.1.1 逻辑型 (logical)

- 逻辑型：仅有 TRUE 和 FALSE 或可写成 T 与 F；用于表述一个描述是否正确

```r
# l1,l2,l3 均为逻辑型变量
l1 <- TRUE
l2 <- FALSE
l3 <- T  # 等价于l1
l4 <- F  # 等价于l2
```

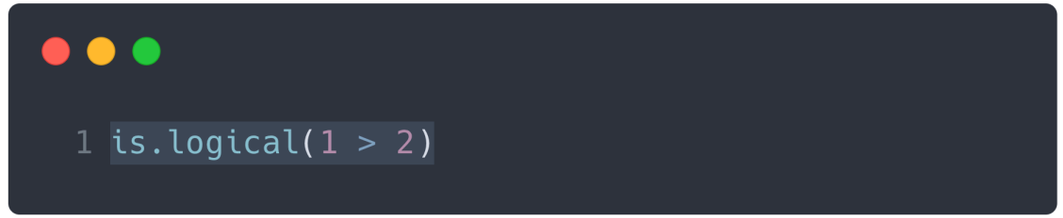
2.1.1 逻辑型 (logical)

- 逻辑型：仅有 TRUE 和 FALSE 或可写成 T 与 F；用于表述一个描述是否正确
- 以下的结果也为逻辑型：
  - `1 > 2`
  - `1 == 0`
  - `1 != 0`

```
1 1 > 2   # 1大于2吗?
2 1 == 0  # 1等于0吗?
3 1 != 0  # 1不等于0吗?
```

2.1.1 逻辑型 (logical)

- 逻辑型：仅有 TRUE 和 FALSE 或可写成 T 与 F；用于表述一个描述是否正确
- 以下的结果也为逻辑型：
  - `1 > 2`
  - `1 == 0`
  - `1 != 0`
- 判断一个变量是否为逻辑型 `is.logical(变量名)`



2.1.2 数值型 (numeric)

- 数值型：最常用的变量类型，代表一个实数。
- **科学计数法** 4e3 相当于 4×10^3

```r
# n1, n2, n3, n4 均为数值型变量。R语言默认存储为浮点型
n1 <- 3
n2 <- 1/3
n3 <- 0.33333
n4 <- 3.3333e-1   # 科学计数法：3.3333e-1 等同于 3.3333 * (10^-1)
```

2.1.2 数值型 (numeric)

- 数值型：最常用的变量类型，代表一个实数。
- **科学计数法** 4e3 相当于 4×10^3
- 逻辑型可以被当成数值型进行计算:
  - TRUE 被转化为 1
  - FALSE 被转化为 0

```
1 TRUE + 1 == 2
2 FALSE + T == 1
```

2.1.3 字符型 (character)

- 代表一个字符串，一般由 " 或 ' 包裹
- 若希望在字符串中引入 " 或 '，可以：
    1. 在外面用另一种引号包裹（如希望在字符串中加入 " 则在外面用 ' 包裹）
    2. 使用转义字符，\" 或 \'

```
c1 = "TRUE"
c2 = "FALSE"
c3 = "this is 'test'"     # 若希望在字符串中加'可以在外侧用"
c4 = 'this is \'test\''   # 也可以在'前加\实现字符的转意
```

2.1.4 不同类型的之间的相互转换

- 转换为数值型：`as.numeric()`
  - 字符型 $\longrightarrow$ 数值型
  - 逻辑型 $\longrightarrow$ 数值型

```r
a <- "123"
as.numeric(a)

b <- T
as.numeric(b)
```

2.1.4 不同类型的之间的相互转换

- 转换为数值型：as.numeric()
- 转换为字符串：as.character()

```r
a <- T
as.character(a)

b <- 1e-3
as.character(b)
```

2.1.4 不同类型的之间的相互转换

- 转换为数值型：`as.numeric()`
- 转换为字符串：`as.character()`
- 转换为逻辑型：`as.logical()`
  - 数值型 $\longrightarrow$ 逻辑型

```r
a <- 1
as.logical(a)

b <- -1
as.logical(b)

c <- 0
as.logical(c)
```

2.1.4 不同类型的之间的相互转换

- 转换为数值型：as.numeric()
- 转换为字符串：as.character()
- 转换为逻辑型：as.logical()
- 对于无法转换的类型，R会自动转化为NA即缺失值

```r
a <- "0"
as.logical(a)

b <- "1this is characters"
as.numeric(b)
```

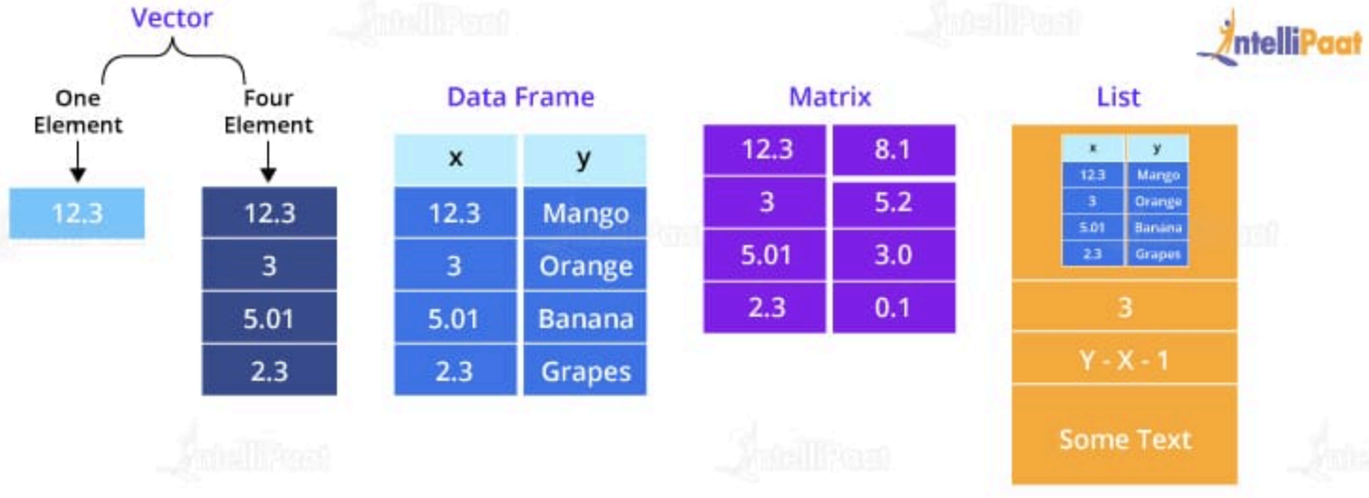
2.1.5 R语言中的常量

变量名	含义
pi	圆周率，3.1415…
Inf	无穷大 1/0
NaN	不定量，0/0
NA	缺失值

03 R的重要数据结构

3.0 重要数据类型

1. 向量 (vector)
2. 矩阵 (matrix)
3. 数据框 (data frame)
4. 列表 (list)



3.1 向量 (vector)

$$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

- 最基本的数据对象，其各分量具有相同的数据类型。
- **向量生成**
    1. c( )

```r
v1 <- c(1,2,3,4)
```

3.1 向量 (vector)

$$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

- 最基本的数据对象，其各分量具有相同的数据类型。
- **向量生成**
    1. `c()`
    2. `s:e`

```
1 v2 <- 1:4
```

3.1 向量 (vector)

$$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

- 最基本的数据对象，其各分量具有相同的数据类型。
- **向量生成**
    1. `c()`
    2. `s:e`
    3. `seq()`

```r
seq(-2, 3, by=0.5) # 公差为 0.5，首项为-2，尾项最接近3的等差数列
seq(2, 4, length.out=4) # 生成首项和尾项分别为2和4的4维向量
seq(2, by=0.5, length=4)  # 生成首项为2，公差为0.5，长度为4的向量
```

3.1 向量 (vector)

$$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

- 最基本的数据对象，其各分量具有相同的数据类型。
- **向量生成**

> **注意**
>
> 当输入元素类型不一致时，R会强制转换；优先级：字符型 > 数值型 > 逻辑型；优先级低的会被强制转换为优先级高的

```r
c(1,2,"3")     # 字符型最优先
c(1,2,TRUE)    # 其次是数值型
```

3.1 向量 (vector)

$$\begin{bmatrix}1 & 2 & 3 & 4\end{bmatrix}$$

- 最基本的数据对象，其各分量具有相同的数据类型。
- **向量生成**
- **元素的索引和更改**
  - a[i]：索引向量 a 中第 i 个元素，其中 i 既可以是单个数值，也可以是由数值或逻辑值组成的向量

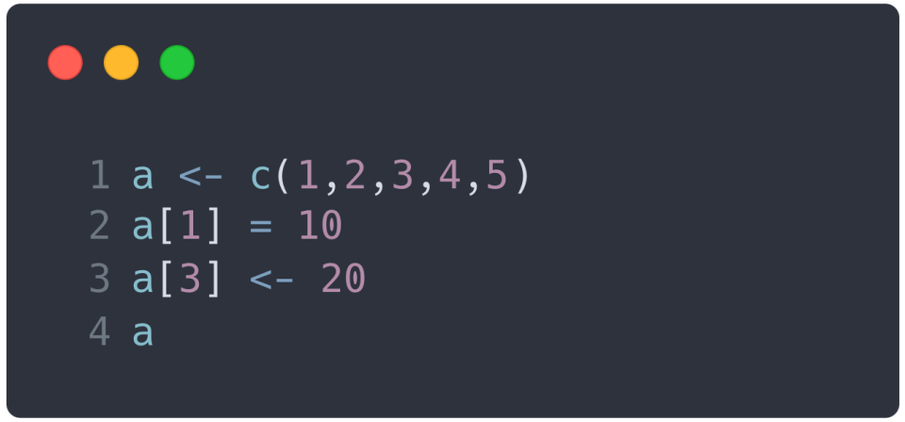
```r
a <- c(1,2,3)
b <- c(2,2,2,1,3)
a[b]

c <- a >= 2    # 返回一个由逻辑值组成的向量
a[c]
```

3.1 向量 (vector)

$$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$$

- 最基本的数据对象，其各分量具有相同的数据类型。
- **向量生成**
- **元素的索引和更改**
  - `a[i]`：索引向量 `a` 中第 `i` 个元素，其中 `i` 既可以是单个数值，也可以是由数值或逻辑值组成的向量
  - `a[i] <- value`(`a[i] = value`)：更改 `a` 中第 `i` 的元素的值

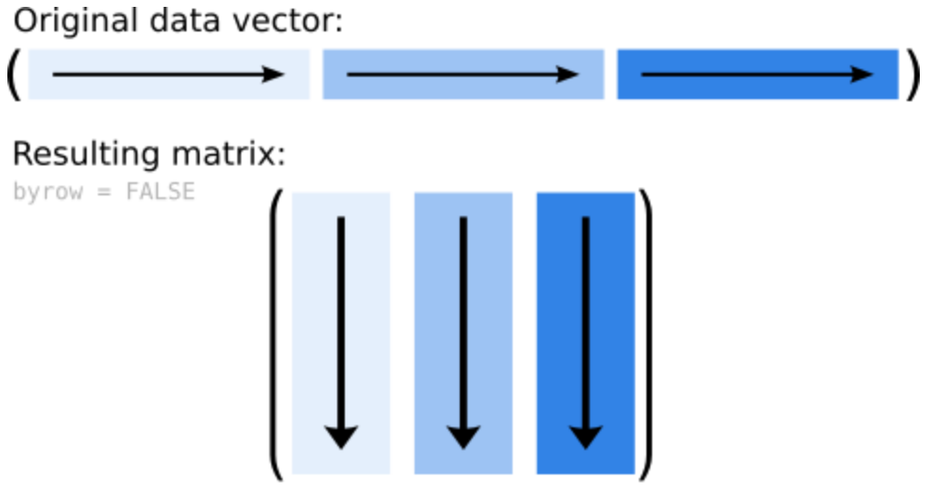

a <- c(1,2,3,4,5)
a[1] = 10
a[3] <- 20
a

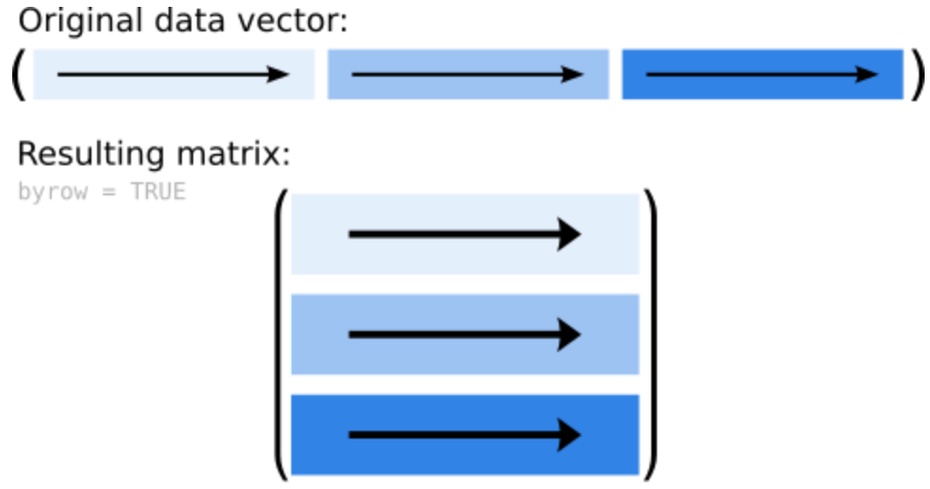
3.2 矩阵 (matrix)

- 矩阵是向量的一种推广形式，和向量一样，其**元素数据类型一致**（因此和向量一样存在数据类型的强制转换）

3.2 矩阵 (matrix)

- 矩阵是向量的一种推广形式，和向量一样，其**元素数据类型一致**
- **矩阵的生成** `matrix(x,n,m,byrow=T/F)` 将向量 x 转化为 m 行 n 列的矩阵





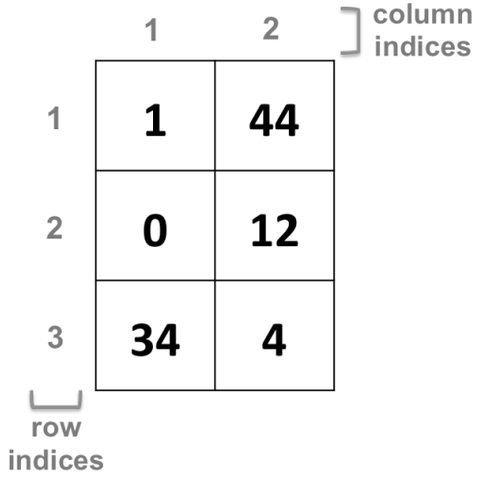

```r
x <- 1:6
m1 <- matrix(x, 3, 2)
```

```r
x <- 1:6
m2 <- matrix(x, 3, 2, byrow=TRUE)
```

3.2 矩阵 (matrix)

- 矩阵是向量的一种推广形式，和向量一样，**其元素数据类型一致**
- **矩阵的生成**
- **矩阵的索引与元素的更改**
  - m[i,j] 索引第 i 行第 j 列的元素；
  - m[i,] 索引第 i 行的所有元素；m[,j] 索引第 j 列的所有元素
  - i, j 也可为向量

```r
m <- matrix(1:9,3,3)
m
m[1,]   # 索引第1行
m[,2]   # 索引第2列
m[1,2]
i <- c(1,3)
j <- c(1,2)
m[i,j]  # 索引第1，3行和第1，2列相交的所有元素
```



3.2 矩阵 (matrix)

- 矩阵是向量的一种推广形式，和向量一样，其**元素数据类型一致**
- **矩阵的生成**
- **矩阵的索引与元素的更改**
  - 更改和向量一样，直接对索引的元素进行赋值即可

```r
m <- matrix(1:9,3,3)
m[1,3] <- 10 # 将第1行第3列的元素改为10
i <- c(1,3)
j <- c(1,2)
# 根据列优先原则，以此将(1,1),(3,1),(1,2),(3,2) 改为 1, 2, 3, 4
m[i,j] <- 1:4
```

3.3 数据框 (data frame)

- 数据框 (data frame) 为矩阵的一种推广：
  - 行表示的是每一个样本的一条数据（一行的数据可能有很多种类型）
  - 列表示的某一个属性所有样本的数据（一列的数据只有一种类型）

	x	y	sex
1	160	51	female
2	175	72	male

3.3 数据框 (data frame)

- 数据框的生成
  - `data.frame(col_name1=vec1, col_name2=vec2, ...)`

```r
x <- c(160, 175)
y <- c(51, 72)
sex <- c("female", "male")
# 使用列变量名（x, y, sex）
df <- data.frame(x,y,sex)
# 设置列变量名分别为(height, weight, sex)
df2 <- data.frame(height=x, weight=y, sex)
```

3.3 数据框 (data frame)

- **数据框的生成**
- **数据框的索引**
  - `names(df)` 返回 `df` 的所有列变量名
  - `df$col_name` 索引 `df` 下 `col_name` 列的所有行，也可用 `df[j]` 的方式索引第 `j` 列下的所有行

```
1 df$x     # 索引变量x列下的所有行
2 df[1]    # 索引第1列，结果同上
```

3.3 数据框 (data frame)

- **数据框的生成**
- **数据框的索引**
  - `names(df)` 返回 `df` 的所有列变量名
  - `df$col_name` 索引 `df` 下 `col_name` 列的所有行，也可用 `df[j]` 的方式索引第 `j` 列下的所有行
  - `df$col_name[i]` 索引 `df` 下 `col_name` 列下的第 `i` 行数据

```
df$x[1]      # 索引变量x列，第1行的元素
df[1,1]      # 索引第1列第1行元素，结果同上
```

3.3 数据框 (data frame)

- **数据框的生成**
- **数据框的索引**
  - `names(df)` 返回 `df` 的所有列变量名
  - `df$col_name` 索引 `df` 下 `col_name` 列的所有行，也可用 `df[j]` 的方式索引第 `j` 列下的所有行
  - `df$col_name[i]` 索引 `df` 下 `col_name` 列下的第 `i` 个样本的数据
  - 我们可以使用与矩阵类似的方式进行索引 `df[i,j]`

3.4 列表 (list)

- 列表属于向量在另一个方向上的拓展，其内部可以存储不同类型的数据，且可以使用键值 (key) 来索引
- **列表的生成** `list(key1=value1, ...)`

```
1 list1 = list(1, "2", T)
2 list1
3 # key值为 x,y,z
4 list2 = list(x=1, y="2", z=T)
5 list2
```

3.4 列表 (list)

- 列表属于向量在另一个方向上的拓展，其内部可以存储不同类型的数据，且可以使用键值 (key) 来索引
- **列表的生成**
- **列表的索引**：
  - 不同于向量，直接通过 `list1[i]` 返回的并不是列表中第 `i` 个值，而是一个小列表，若希望返回值，使用 `list1i`
  - 类似于数据框，可以使用键值进行索引

```
1 class(list1[1])
2 class(list1[[1]])
3
4 list2$x
5 list2[[1]]  # 结果等价于 list2$x
```

04 基础运算和函数

4.1 基础运算

- 加减乘除：+ − * /
- 幂：^ ** （二者相同）

```
1 1 + 2 * 10     # 乘除以的优先级高于加减
2 (1 + 2) * 10   # () 可以提高计算的优先级
3 2 ^ 3          # 2的3次方
4 2 ** 3         # 2的3次方, 同上
```

4.2 常用函数

- 指数函数 `exp(x)` e^x
- 对数函数 `log(x)` $\ln(x)$; `log(x,base=n)` $\log_n(x)$
- 三角函数 `sin(x)` `cos(x)` `tan(x)`
- 求和 `sum` `colSums` `rowSums`

```r
exp(1)
log(exp(1))
log(2^3, base=2) # 以2为底，8的对数
sin(pi/6)

x <- 1:9
sum(x)
m <- matrix(x,3,3)
colSums(m)
rowSums(m)
```