2 Discovering R

- 学习 R 最好的方式是练习。
- 经过一段时间的 练习 ，最好通过在真实世界中对数据进行一系列预先指定的操作而更好地掌握R。

> 你可以进一步的阅读 Hadley Wickham and Garrett Grolemund's book R for Data Science：R for Data Science (2e)

S：练习（+真实世界的练习）+ 看书

▼ 本章节，我们将对R语言进行一系列的探索：

在这一章中，我们开始进入R宇宙的旅程。这可能是您第一次接触编程，您可能会感到有点焦虑。这是可以理解的，但没有理由担心。在过去的二十年里，世界各地成千上万的聪明人贡献了各种方法，让R的用户更容易、更方便地使用它。我们还将了解一个非常强大的计算机程序（Rstudio），我们可以使用它来编写和运行R代码，使其变得不那么麻烦。cumbersome

然而，与您以前使用过的其他数据分析程序相比，使用R仍然比较困难。R社区最重要的人物之一Hadley Wickham曾经强调，R与基于图形用户界面（GUI）的统计软件有着根本的不同（Grolemund 2014，前言）。**gui允许您通过单击几个按钮来执行数据分析，但是您最终受到开发人员认为重要的功能的限制**。

另一方面，**R没有这些限制**，但可能需要更多的背景知识。像任何语言一样，R需要学习，需要练习才能成为熟练的用户。挫折是令人不快的，但也是这个过程中很自然的一部分。在前言中，你可以找到一个完整的部分，我们描述了一些你可以做的事情，如果你卡住了（最好的现在是问ChatGPT）。

首先，我们要明白，R 不是一个 图形用户界面 。

> Graphical User Interface（GUI，图形用户界面）是一种让用户通过图形方式（而不是纯文本命令）与计算机或电子设备进行交互的界面，数据分析通常通过点击一些按钮即可完成，但功能受限于界面提供的选项（开发者认为重要的功能）。

其次，R语言绝对是值得学习的（理由太多不赘述）。

S：R灵活且功能强大，但难度较大，需要学习。

▼ 安装 R 和 Rstudio

网络发达，不再赘述安装方法。

1. 第一次打开 R Studio 时，它可能看起来很像图2.1中所示。R Studio 中有三个窗格。在右上角，我们有“**环境**”窗格，它显示我们在 *R* 内部定义（即保存）的对象。在右下角，您可以找到“**文件**”、“**绘图**”、“**包**”和“**帮助**”窗格。此窗格具有多种功能：例如，它用于显示计算机上的文件、显示绘图和已安装的包以及访问帮助页面。然而，R Studio 的核心是左侧，您可以在其中找到**控制台**。控制台是输入所有 *R* 代码然后运行的地方。

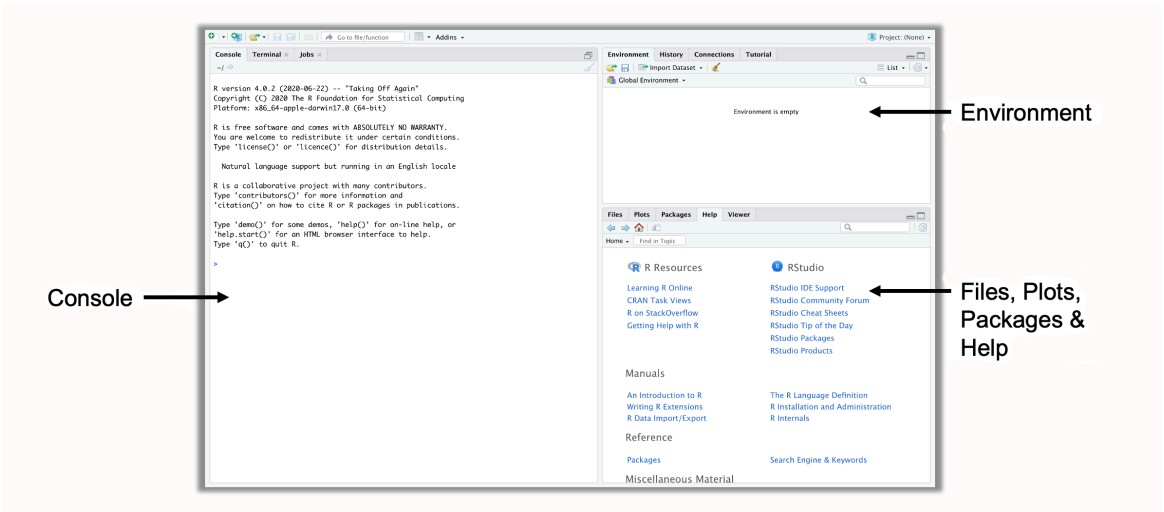


图 2.1：R Studio 中的窗格。

▼ Packages（开始实践部分）

首先，通过 `R code` 安装一些 `R packages`。

> `packages` 的存在是R为什么这么强大的原因之一。简单来说：R的核心是package，package的核心是 `function` ，这允许我们使用 `function` 对我们的数据进行一系列`预先定义的操作`。

在下面的实操中，我们根据这个逻辑反推：先确定我们预先指定的操作是什么，然后确定什么function可以实现这个操作，最后找到内含这个function的package并安装和载入，输入数据和相应的参数给function，function返回我要的结果。

▼ function 的运作方式与使用R的底层逻辑

R 中 function的定义和数学公式保持一致。对 function 进行编码时，首先写下它的名称，然后加上括号，括号中包含函数的`输入`和/或`参数`：

$$f(x)$$

1. 我们预先指定我们想进行的分析：9 的平方根是什么？
2. 接着，我们找到可实现该功能的函数：$\mathrm{sqrt}()$
3. 只需要提供 9 作为 function 的输入就可以得到结果。
4. function 返回我们指定操作的结果。

```
## 1
sqrt(9)

## output
[1] 3
```

总的来说，所有 function 遵循同样的底层逻辑：你提供 function 需要的 parameter 和/或 input，function 为你执行你所期望的计算，并带回相应的output给你。

▼ 安装与载入 packages（确认为我们执行预先指定的操作的相应function包含在哪个/些R packages 后）

首先，安装一些 helpful 的 R packages：之前先确定我们预先指定的操作是什么

函数： `install.packages`

1. tidyverse（我要进行数据整理和可视化）

> 当我们安装 {tidyverse} 包时，它同时为我们提供了 {ggplot2} 、 {dplyr} 、 {tidyr} 、 {readr} 、 {purrr} 、 {stringr} 和 {forcats} 包。

2. meta（我要进行meta分析）
3. metafor（我要进行meta分析）
4. dmetar（我要进行meta分析）

```
install.packages("tidyverse")
install.packages("meta")
install.packages("metafor")
install.packages("devtools")
devtools::install_github("MathiasHarrer/dmetar")
```

⚠️ Do not forget to put the package names into quotation marks (" "). Otherwise, you will get an error message.

然后，将 R packages 载入进环境中：

函数： `library()`

```
library(tidyverse) ## manipulate and visualize data
library(meta)      ## Doing meta analysis
library(metafor)   ## Doing meta analysis
library(dmetar)    ## Doing meta analysis
```

▼ 数据准备与导入（为可实现预先指定操作的function提供`输入`）

本章关注如何使用 R Studio 将数据导入 R：

> 你必须明白，数据准备是乏味和耗时的，但你必须经历，因为其对后续的步骤至关重要。

将数据导入 R 前，需要正确的在 Excel 中将数据进行准备：

将要被导入 R 的数据储存在 Excel 中，以下是一些注意事项，当在Excel中准备数据时：

1. 对数据列进行命名，R会自动将首行内容识别为变量名。
2. 变量名中不能有空格，请使用 _ 或 . （例如 “column_name”）。
3. 变量的排列顺序不重要，只需要确保它们被正确地标记。
4. 删除过去写入过数据的列或行，因为R可能会将其识别为有（缺失）数据存在而导入它们。

在Excel中准备一份规范的数据至关重要，这可以确保下游分析的顺利进行。

让我们从一个示例数据集开始（上面部分的实际操作）：

假设你计划对 自杀预防项目 进行元分析。在你的研究中，你想要关注的 结局指标 是 自杀意念的严重程度 （即个人认为、考虑或计划结束生命的程度），通过问卷进行评估。你已经完成了文献筛选和 数据提取，现在想要在 R 中导入 meta分析数据 。

接下来的任务是将包含所有相关数据的 Excel 表格准备好：

表 2.1 展示了我们想要导入进 R 的所有数据。在第一行，该表还展示了根据我们 上面列出的规则，如何在 Excel 文件中为各变量列命名。

> 上面列出的规则 ：变量名中不能有 空格，请使用 _ 或 . （例如 “column_name”）。

我们可以看到（准备什么内容）：

- 该电子表格将每项研究列在一行中；
- 对于每项研究，干预组和对照组的样本量（n）、平均值和标准差（SD）都包含在内。这是计算效应量所需的结果数据，**我们将在第 3 章详细讨论**；

> 第3章：https://www.notion.so/5-28-5-30-3-Effect-size-1f7980f4c6e380c7807dfb2cd7c915b3?source=copy_link

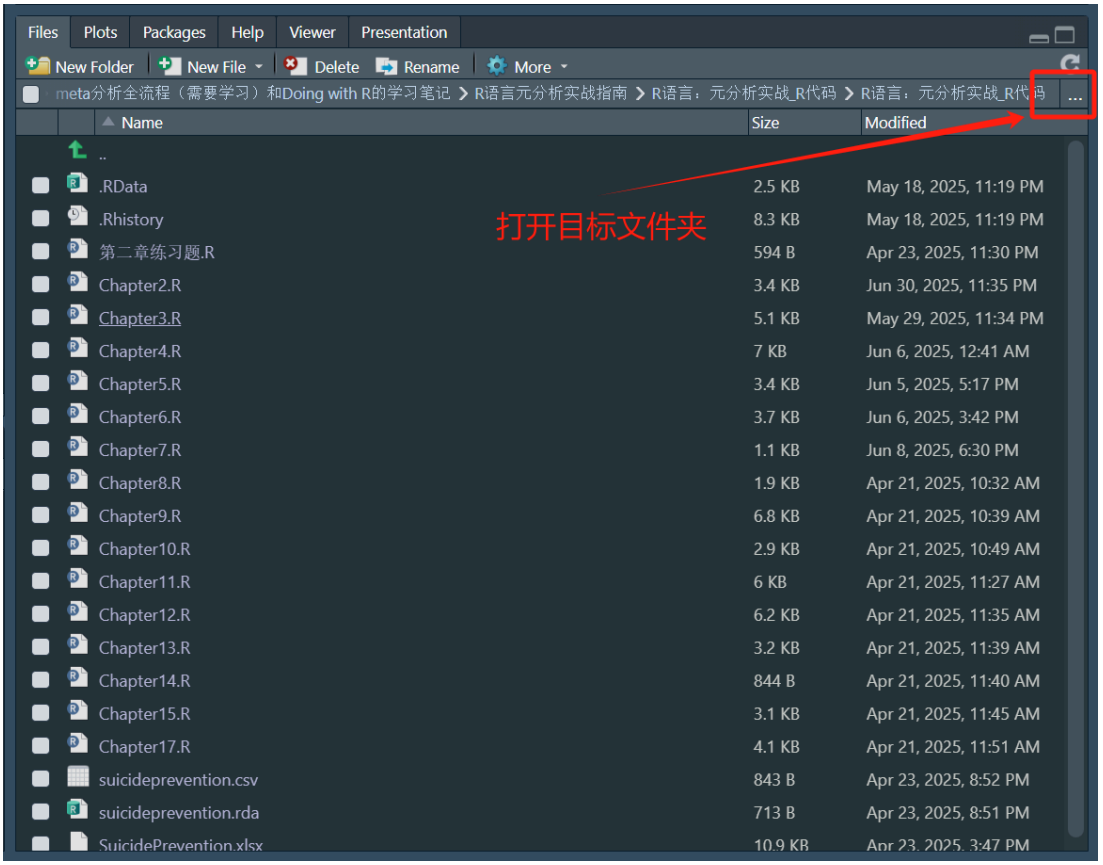
- 接下来的三列包含我们稍后作为元分析一部分想要分析的变量（调节变量 or 亚组分析）。

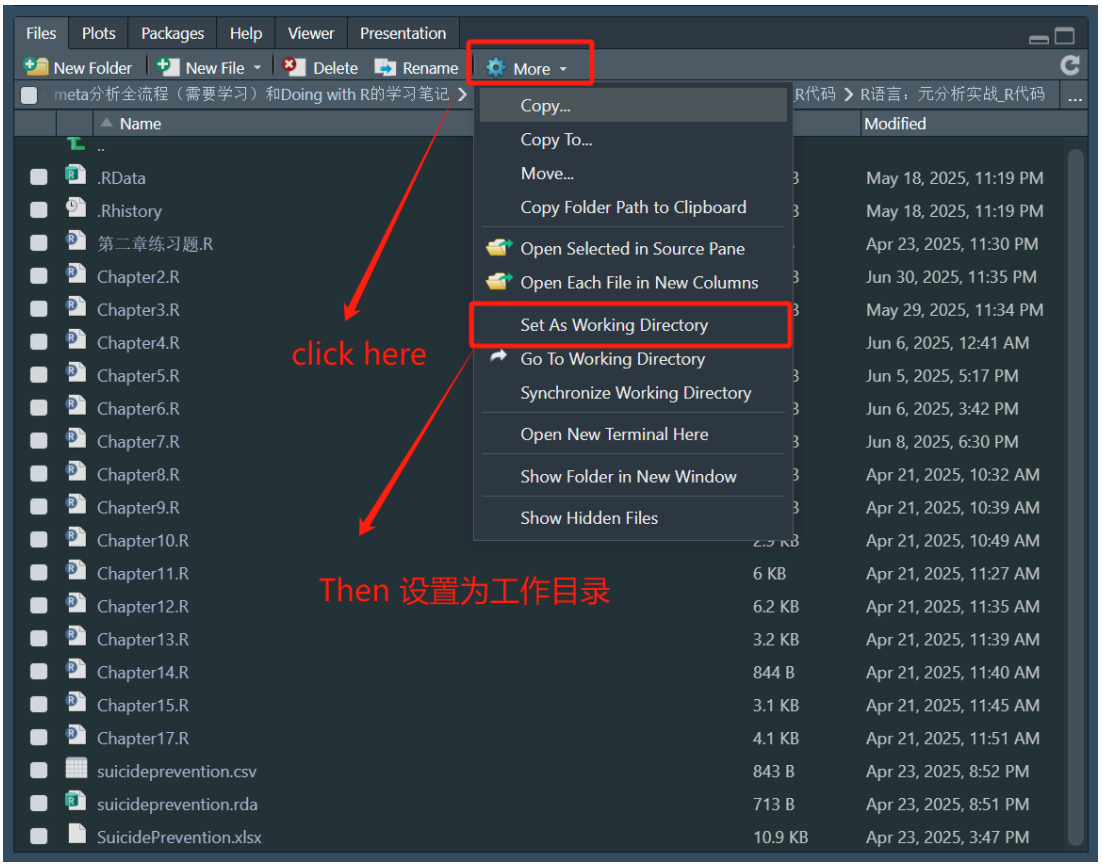
'author'	'n.e'	'mean.e'	'sd.e'	'n.c'	'mean.c'	'sd.c'	'pubyear'	'age_group'	'control'
	Intervention Group			Control Group			Subgroups		
Author	N	Mean	SD	N	Mean	SD	Year	Age Group	Control Group
Berry et al.	90	14.98	3.29	95	15.54	4.41	2006	general	WLC
DeVries et al.	77	16.21	5.35	69	20.13	7.43	2019	older adult	no intervention
Fleming et al.	30	3.01	0.87	30	3.13	1.23	2006	general	no intervention
Hunt & Burke	64	19.32	6.41	65	20.22	7.62	2011	general	WLC
McCarthy et al.	50	4.54	2.75	50	5.61	2.66	1997	general	WLC
...	...	...	...	...	...	...	...	...	...

表 2.1: The suicide prevention dataset.

当Excel 表格准备好后，我们想将数据导入 R ，首先需要设置 工作目录 ，也就是一个文件夹，所有的 原始数据 和 输出结果 都被保存在里面，以保证 R 可以读取里面的数据进行分析并输出结果进里面。

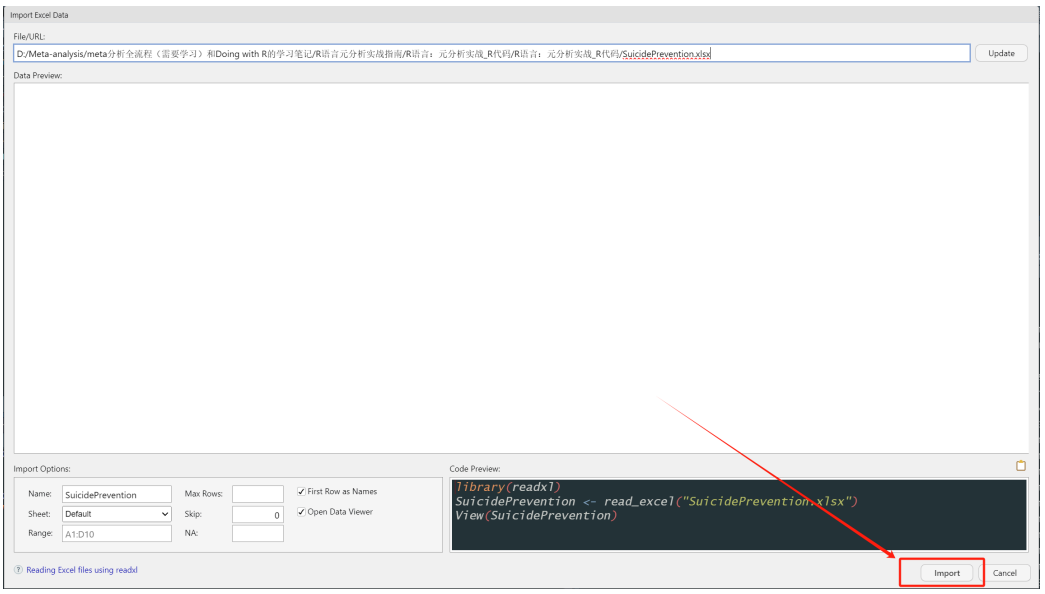
设置工作目录的方法是：


Files
Plots
Packages
Help
Viewer
Presentation
New Folder
New File
Delete
Rename
More
meta分析全流程（需要学习）和Doing with R的学习笔记 > R语言元分析实战指南 > R语言：元分析实战_R代码 > R语言：元分析实战_R代码
Name
Size
Modified
打开目标文件夹
.RData 2.5 KB May 18, 2025, 11:19 PM
.Rhistory 8.3 KB May 18, 2025, 11:19 PM
第二章练习题.R 594 B Apr 23, 2025, 11:30 PM
Chapter2.R 3.4 KB Jun 30, 2025, 11:35 PM
Chapter3.R 5.1 KB May 29, 2025, 11:34 PM
Chapter4.R 7 KB Jun 6, 2025, 12:41 AM
Chapter5.R 3.4 KB Jun 5, 2025, 5:17 PM
Chapter6.R 3.7 KB Jun 6, 2025, 3:42 PM
Chapter7.R 1.1 KB Jun 8, 2025, 6:30 PM
Chapter8.R 1.9 KB Apr 21, 2025, 10:32 AM
Chapter9.R 6.8 KB Apr 21, 2025, 10:39 AM
Chapter10.R 2.9 KB Apr 21, 2025, 10:49 AM
Chapter11.R 6 KB Apr 21, 2025, 11:27 AM
Chapter12.R 6.2 KB Apr 21, 2025, 11:35 AM
Chapter13.R 3.2 KB Apr 21, 2025, 11:39 AM
Chapter14.R 844 B Apr 21, 2025, 11:40 AM
Chapter15.R 3.1 KB Apr 21, 2025, 11:45 AM
Chapter17.R 4.1 KB Apr 21, 2025, 11:51 AM
suicideprevention.csv 843 B Apr 23, 2025, 8:52 PM
suicideprevention.rda 713 B Apr 23, 2025, 8:51 PM
SuicidePrevention.xlsx 10.9 KB Apr 23, 2025, 3:47 PM


Files
Plots
Packages
Help
Viewer
Presentation
New Folder
New File
Delete
Rename
More
Copy...
Copy To...
Move...
Copy Folder Path to Clipboard
Open Selected in Source Pane
Open Each File in New Columns
Set As Working Directory
Go To Working Directory
Synchronize Working Directory
Open New Terminal Here
Show Folder in New Window
Show Hidden Files
click here
Then 设置为工作目录

1. **GUI操作**

Then，点击相应的Excel，点击import dataset...紧接着import：



2. **也可使用代码块进行导入，代码如下：**

```
library(openxlsx)
SuicidePrevention ← read.xlsx("SuicidePrevention.xlsx")
View(SuicidePrevention)
```

然后，数据集应该在右上角的 `Environment` 中以“`SuicidePrevention`”的名称列出，这意味着你的数据现在已经被加载进R环境，并且可以被R代码调用。

我们在这里导入的表格数据集在R中称为数据框（`data.frame`），其包含行与列，类似于`Excel`的`Spreadsheet`。

对于一个很棒的分析，应该将教程流程进行反推，明白所有的前置条件是什么，然后再推回去：

1. Excel准备：内容是什么（以便有可用 `输入` 执行预先指定的操作）；数据格式是什么？（成功导入数据的前提）
2. 导入数据：工作目录的设置。
3. **数据操作（下面）**
4. 确定预先制定的操作，找到实现这个操作的function，最后找到内含这个function的package并安装和载入，输入数据（先前准备并导入的Excel数据）和相应的参数给function，function返回我要的结果。

▼ 数据操作

指（使用 `Tidyverse` 狗头）对数据进行各种处理、转换、整合等操作的过程。

已经做了什么？

我们对数据进行了准备（根据 `R` 所定义的规则在 `Excel` 中准备数据）

我们将数据导入进了 `R` 作为 `data.frame` 待后续使用。

现在我们要做的是 —— 对导入进 `R` 的数据进行一系列的数据操作：

> 数据整理（Data wrangling）：将数据从一种原始数据形式转换和映射到另一种格式的过程，旨在使其更适合于各种下游用途，如可视化和分析。

科学家会花费大量的时间在数据整理上，即将untidy的数据转换为tidy的数据，以便更好的进行后续的分析与可视化。

如何实现 Data wrangling?

R包 `{tidyverse}` 的 function 为数据整理提供了一个出色的 `工具箱` 。

▼ 2.5.1 数据类别的转换

当将数据导入后，我们第一步应该做什么?

观测数据（and Tidy your data），使用 `{tidyverse}` 包的 `glimpse()` function looks like：（其实 `str()` 更好(*/ω＼*)）

```
glimpse(SuicidePrevention)
```

```
> glimpse(SuicidePrevention)
Rows: 9
Columns: 10
$ author     <chr> "Berry et al.", "DeVries et al.", "Fleming et al.", "Hunt & Bur…
$ n.e        <chr> "90", "77", "30", "64", "50", "109", "60", "40", "51"
$ mean.e     <chr> "14.98", "16.21", "3.01", "19.32", "4.54", "15.11", "3.44", "7.…
$ sd.e       <chr> "3.29", "5.35", "0.87", "6.41", "2.75", "4.63", "1.26", "0.76",…
$ n.c        <dbl> 95, 69, 30, 65, 50, 111, 60, 40, 56
$ mean.c     <chr> "15.54", "20.13", "3.13", "20.22", "5.61", "16.46", "3.42", "7.…
$ sd.c       <chr> "4.41", "7.43", "1.23", "7.62", "2.66", "5.39", "1.88", "1.41",…
$ pubyear    <dbl> 2006, 2019, 2006, 2011, 1997, 2000, 2013, 2015, 2014
$ age_group <chr> "general", "older adult", "general", "general", "general", "gen…
$ control    <chr> "WLC", "no intervention", "no intervention", "WLC", "WLC", "no …
>
```

现在，我们看到，`glimpse` function为我们提供了存储在数据集中的每一列数据类型的详细信息。有不同的缩写表示不同类型的数据。在R语言中，它们被称为 `类` ：

`<num>` 代表 **数值型**。这是所有以数字形式存储的数据（例如：1.02）。

`<chr>` 代表 **字符型**。这是所有以文字形式存储的数据。

`<log>` 代表 **逻辑型**。这些变量是二元的，意味着它们表示某个条件是否成立，其取值为 `TRUE` 或 `FALSE` 。

`<factor>` 代表 **因子型**。因子是以数字形式存储的，每个数字表示变量的一个不同水平。例如，变量的可能因子水平可以是：`1 = “低”` 、 `2 = “中”` 、 `3 = “高”` 。

我们还可以使用 `class` function 来检查列的类别。我们可以通过在列名前加上 $ operator 直接访问数据框中的列，然后加上列名。让我们来试一下。首先，让 R 为我们提供列 n.e 中的数据。之后，我们检查该列的类别：

```
SuicidePrevention$n.e
## [1] "90"  "77"  "30"  "64"  "50"  "109" "60"  "40"  "51"

class(SuicidePrevention$n.e)
## [1] "character"
```

我们看到，包含干预组的样本量的列 n.e 具有 "`character`" 特征。但是等等，这是错误的类！在导入过程中，该列被错误地归类为字符变量，而它实际上应该是 “numeric”。这对进一步的分析步骤有影响。例如，如果我们想计算平均样本量，我们会得到这样的警告：

```
> mean(SuicidePrevention$n.e)
[1] NA
警告信息:
In mean.default(SuicidePrevention$n.e) : 参数不是数值也不是逻辑值：返回NA
```

咋办！？

为了使我们的数据集可用，我们通常在观测完数据后必须先将列转换为正确的类：（也就是str()之后立马转换数据储存形式，省的后面一堆麻烦事儿，就像上面...）

我们可以使用一组都以 `“as.”` 开头的函数：

- `as.numeric`
- `as.character`
- `as.logical`
- `as.factor`

让我们来看几个例子：

- 在之前 `glimpse` function 的输出中，我们看到有几列被赋予了 character 类型，而它们应该是numeric 类型的；
- 我们看到出版年份 pubyear 具有类 `<dbl>` 。这代表双精度，意味着列是一个数字向量。在R中使用 double 和 numeric 来引用数值数据类型是一个历史遗留问题。然而，这通常没有实际的差异，`无需在意`。

注意，`类`被错误的标记可能会导致下游问题，怎么办呢？

我们在此例中使用 `as.numeric()` function 将类别转换为 numeric：

我们为函数提供要更改的列，然后使用赋值操作符（←）将输出保存回原始位置。这将生成以下代码：

> 在前面的章节已经提到，我们想执行预先计划的分析，只需要先找到可实现该功能的 function，确定其属于哪个 R 包，安装和加载，调用该函数，接着给该函数进入参数和信息即可。

```
SuicidePrevention$n.e ← as.numeric(SuicidePrevention$n.e)
SuicidePrevention$mean.e ← as.numeric(SuicidePrevention$mean.e)
SuicidePrevention$sd.e ← as.numeric(SuicidePrevention$sd.e)
SuicidePrevention$n.c ← as.numeric(SuicidePrevention$n.c)
SuicidePrevention$mean.c ← as.numeric(SuicidePrevention$mean.c)
SuicidePrevention$sd.c ← as.numeric(SuicidePrevention$sd.c)
```

为了进行亚组分析，将变量编码为因子类型：

我们还可以在输出中看到 age_group 和 control（数据中的子组）被编码为字符。然而，在现实中，将它们编码为因子更合适，每个因子有两个水平。我们可以用 `as.factor` 函数来改变类：

```
SuicidePrevention$age_group ← as.factor(SuicidePrevention$age_group)
```

```
SuicidePrevention$control ← as.factor(SuicidePrevention$contr
ol)
```

使用 `levels` 和 `nlevels` function，我们还可以查看`因子标签`和`因子中的层数`：

```
levels(SuicidePrevention$age_group)
## [1] "general"     "older adult"

nlevels(SuicidePrevention$age_group)
## [1] 2
```

通过 `levels` function 改变因子的 `标签` ：

在R中实现该目的，需要使用 `c` funtion，这个 function 可以将两个或多个单词或数字连接在一起并创建一个元素。**让我们试一下：**

1. 先将新的因子标签储存进一个元素中：

```
new.factor.levels ← c("gen", "older")
new.factor.levels
```

2. 很好，现在可以使用新创建的 `new.factor.levels` 对象，并将其分配给 `age_group` 列的因子标签：

```
levels(SuicidePrevention$age_group) ← new.factor.level
s
```

3. 让我们检查一下重命名是否成功：

```
SuicidePrevention$age_group
## [1] gen   older gen   gen   gen   gen   gen   older older
## Levels: gen older
```

注意，对 factor 进行新的 level 赋值时，一定要先将变量列转换为 factor 类别进行储存。

为了进行亚组分析，将变量编码为逻辑类型：

还可以使用 `as.logical` 创建逻辑。假设我们想重新编码 `pubyear` 列，以便它只显示2009年之后发表的研究。要做到这一点，我们必须通过代码定义一个是/否规则。我们可以使用“大于或等于”操作符 `>=` 来完成此操作，然后将其用作 `as.logical` function 的输入。

```
SuicidePrevention$pubyear
as.logical(SuicidePrevention$pubyear >= 2010)
## [1] FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TR
UE
SuicidePrevention$pubyear.factor ← as.logical(SuicidePreventio
```

```
n$pubyear >= 2010)
class(SuicidePrevention$pubyear.factor)
SuicidePrevention$pubyear.factor ← as.factor(SuicidePrevention
$pubyear.factor)
levels(SuicidePrevention$pubyear.factor)
nlevels(SuicidePrevention$pubyear.factor)
new.pubfac.levels ← c("less than 2010", "more than 2010")
new.pubfac.levels
levels(SuicidePrevention$pubyear.factor) ← new.pubfac.levels
levels(SuicidePrevention$pubyear.factor)
```

▼ 2.5.2 数据取子集

The `$` operator 可以取 `data.frame` 的 `column`

但是：

the square bracket 也很常用： `data.frame[rows, columns]` ，仅使用 dataframe 特定的行和列参数，即可取出相应的数据子集：

```
SuicidePrevention[2,]
##      author n.e mean.e sd.e n.c mean.c sd.c pubyear age_grou
p         control
##  2 DeVries et al.  77  16.21 5.35  69  20.13 7.43    2019     older
no intervention
```

我们可以更具体地告诉R我们只想要第二行第一列的信息：

```
SuicidePrevention[2, 1]
## [1] "DeVries et al."
```

为了选择特定的切片，我们必须再次使用 `concatenate (c)` 函数。例如，如果我们想提取第2行和第3行以及第4列和第6列，我们可以使用以下代码：

```
SuicidePrevention[c(2,3), c(4,6)]
##   sd.e mean.c
## 2 5.35  20.13
## 3 0.87   3.13
```

对于取列子集，我们不仅可以提供列序号，通常情况下，我们知道列的相应变量名，直接在列的参数位置输入列名进行相应提取：

```
SuicidePrevention[, c("author", "control")]
##           author        control
## 1    Berry et al.            WLC
## 2  DeVries et al. no intervention
## 3  Fleming et al. no intervention
## 4    Hunt & Burke            WLC
## 5 McCarthy et al.            WLC
## 6   Meijer et al. no intervention
## 7   Rivera et al. no intervention
## 8  Watkins et al. no intervention
## 9  Zaytsev et al. no intervention
```

另一种可能性是根据行值筛选数据集。我们可以使用 filter 函数来做到这一点。在函数中，我们需要指定数据集名称，以及filter 的逻辑。一个相对简单的例子是filter 所有n.e 等于或小于50 的研究：

```
filter(SuicidePrevention, n.e <= 50)
##            author n.e mean.e sd.e n.c mean.c sd.c pubyear age_g
roup
## 1  Fleming et al.  30   3.01 0.87  30   3.13 1.23    2006      gen
## 2 McCarthy et al.  50   4.54 2.75  50   5.61 2.66    1997       ge
n
## 3  Watkins et al.  40   7.10 0.76  40   7.38 1.41    2015     older
##           control
## 1 no intervention
## 2             WLC
## 3 no intervention
```

但也可以按名字 filter。假设我们想要提取 **Meijer** 和 **Zaytsev** 两位作者的研究。为此，我们必须使用 `%in%` operator 和 `c` 函数定义 filter 逻辑：

```
filter(SuicidePrevention, author %in% c("Meijer et al.",
"Zaytsev et al."))
##          author n.e mean.e sd.e n.c mean.c  sd.c pubyear age_g
roup
## 1  Meijer et al. 109  15.11 4.63 111  16.46  5.39    2000      gen
## 2 Zaytsev et al.  51  23.74 7.24  56  24.91 10.65    2014    older
##          control
## 1 no intervention
## 2 no intervention
```

不想看见他俩也行，只需要添加 `!`：

```
filter(SuicidePrevention, !author %in% c("Meijer et al.",
"Zaytsev et al."))
```

▼ 2.5.3 数据替换（先索引，后赋值）

在 `R` 中可以对特定索引的数据进行切换，why we dont use the excel spreadsheet to change those of values?

```
SuicidePrevention[2, "pubyear"] ← 2018
SuicidePrevention[2, "pubyear"]
```

我们索引位置 使用 assignment 符 `←` 进行数据替换。

统一对数据集进行批量操作：

```
SuicidePrevention$mean.e + 5
## [1] 19.98 21.21  8.01 24.32  9.54 20.11  8.44 12.10 28.74
```

Do caculations：（其实使用metafor escalc func就可以）

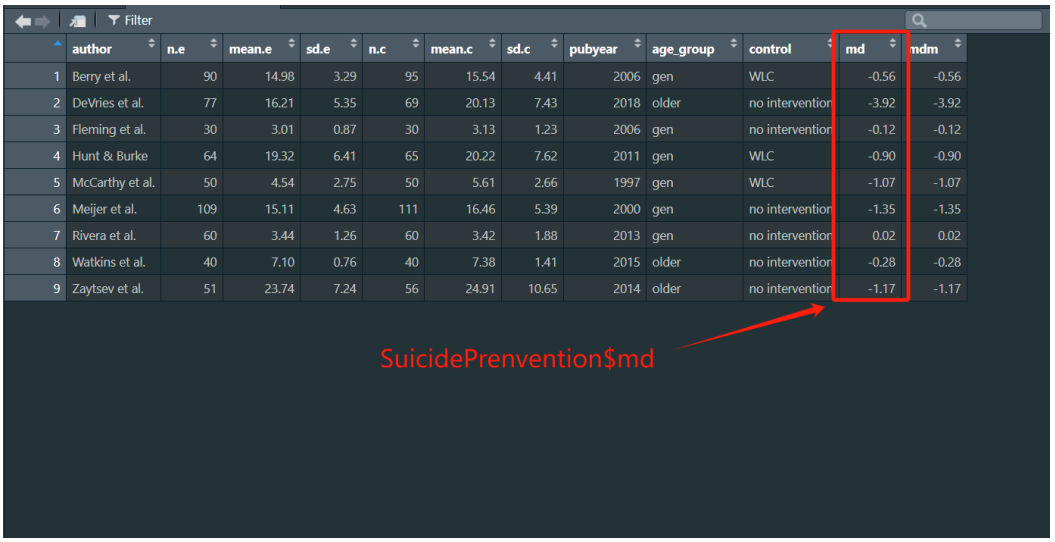
md：

```
SuicidePrevention$mean.e - SuicidePrevention$mean.c
## [1] -0.56 -3.92 -0.12 -0.90 -1.07 -1.35  0.02 -0.28 -1.17
```

we want to use this mean difference later on, we need to save it as an extra data column called md：

```
md ← SuicidePrevention$mean.e - SuicidePrevention$mean.c
```

```
SuicidePrevention$md ← SuicidePrevention$mean.e - SuicidePr
evention$mean.c
```



	author	n.e	mean.e	sd.e	n.c	mean.c	sd.c	pubyear	age_group	control	md	mdm
1	Berry et al.	90	14.98	3.29	95	15.54	4.41	2006	gen	WLC	-0.56	-0.56
2	DeVries et al.	77	16.21	5.35	69	20.13	7.43	2018	older	no intervention	-3.92	-3.92
3	Fleming et al.	30	3.01	0.87	30	3.13	1.23	2006	gen	no intervention	-0.12	-0.12
4	Hunt & Burke	64	19.32	6.41	65	20.22	7.62	2011	gen	WLC	-0.90	-0.90
5	McCarthy et al.	50	4.54	2.75	50	5.61	2.66	1997	gen	WLC	-1.07	-1.07
6	Meijer et al.	109	15.11	4.63	111	16.46	5.39	2000	gen	no intervention	-1.35	-1.35
7	Rivera et al.	60	3.44	1.26	60	3.42	1.88	2013	gen	no intervention	0.02	0.02
8	Watkins et al.	40	7.10	0.76	40	7.38	1.41	2015	older	no intervention	-0.28	-0.28
9	Zaytsev et al.	51	23.74	7.24	56	24.91	10.65	2014	older	no intervention	-1.17	-1.17

▼ 2.5.4 Pipe operator

In R，Pipe are written as `%>%`

> Pipes essentially allow us to apply a function to an object `without having to specify the object name directly in the function call`.

举个栗子，caculate mean sample size：

```
SuicidePrevention$n.c %>%
mean()
## [1] 64
```

The special strength of pipes comes from the fact that `they allow us to chain many functions together`.
求大于2009年以后研究的 squre root of the mean control sample size

Pipes let us do this conveniently in one step：

```
## 管道符的chain操作，集成一系列的function call
SuicidePrevention %>%
  filter(pubyear > 2009) %>%
  pull(n.c) %>%
  mean() %>%
  sqrt()
  ## [1] 7.615773
```

> the `pull` function. This function can be seen as an equivalent of the `$` operator that we can use in pipes.

It simply “pulls out” the variable we specify in the function like `n.c` ，这会使得管道下游调用我们pull后pipe的information。

Accessing the *R* Documentation

it is `impossible` to remember how all functions are used correctly!!!

```
查看帮助文档
?theme
```

The *R* documentation of a function usually at least contains a `Usage`, `Arguments` and `Examples` section. 有助于理解特定 `function call` 的用法。

▼ 2.5.4 Save

1. 保存.rda

```
save(SuicidePrevention, file = "suicideprevention.rda")
```

2. 写出.csv（方便下次直接load environment中的数据）

```
write.csv(SuicidePrevention, file = "suicideprevention.csv")
```

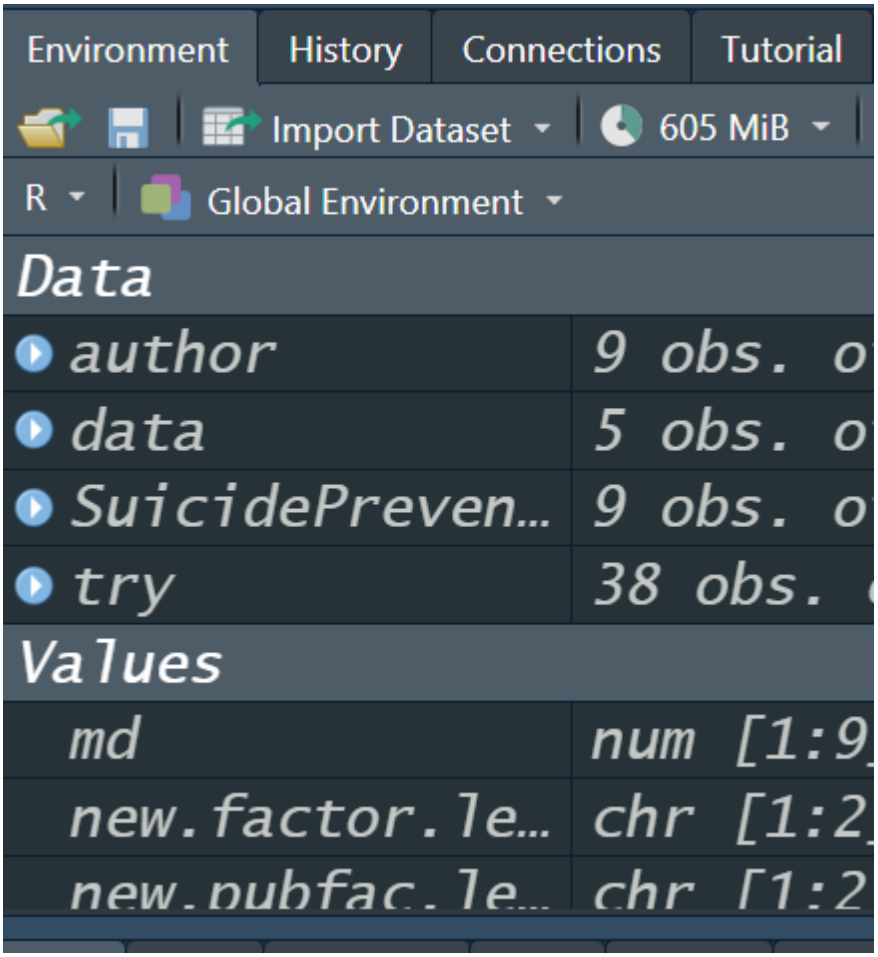
3. 保存excel

```
library(writexl)

## 创建示例数据
df ← data.frame(
Name = c("Tom", "Lucy", "Jack"),
Score = c(90, 85, 88)
)

write_xlsx(df, "output.xlsx") ## 保存为 Excel 文件
```

4. 保存 R workspace



▼ Practice more

首先创建一个模拟dataset用于练习数据操作：

```
#data manipulation exercises
data ← data.frame("Author" = c("Jones", "Goldman", "Townsend",
"Martin", "Rose"),
                  "TE" = c(0.23, 0.56, 0.78, 0.23, 0.33),
                  "seTE" = c(0.324, 0.235, 0.394, 0.275, 0.348),
                  "subgroup" = c("one", "one", "two", "two", "three"))
```

瞥一眼生成的dataset：

```
> glimpse(data)
Rows: 5
Columns: 4
$ Author   <chr> "Jones", "Goldman", "Townsend", "Martin", "Rose"
$ TE       <dbl> 0.23, 0.56, 0.78, 0.23, 0.33
$ seTE     <dbl> 0.324, 0.235, 0.394, 0.275, 0.348
$ subgroup <chr> "one", "one", "two", "two", "three"
```

Here are the exercises for this data set.

1. Show the variable `Author`.
2. Convert `subgroup` to a factor.
3. Select all the data of the “Jones” and “Martin” study.
4. Change the name of the study “Rose” to “Bloom”.
5. Create a new variable `TE_seTE_diff` by subtracting `seTE` from `TE`. Save the results in `data`.
6. Use a pipe to (1) filter all studies in `subgroup` “one” or “two”, (2) select the variable `TE_seTE_diff`, (3) take the mean of the variable, and then apply the `exp` function to it. Access the *R* documentation to find out what the `exp` function does.

▼ Summary

- *R* has become one of the most powerful and frequently used statistical programming languages in the world.
- *R* is not a computer program with a graphical user interface and pre-defined functionality. It is a full programming language to which people all around the world can contribute freely available add-ons, so-called **packages**.
- R Studio is a computer program which allows us to use *R* for statistical analyses in a convenient way.
- The fundamental building blocks of *R* are functions. Many of these functions can be imported through packages which we can install from the internet.
- Functions can be used to import, manipulate, analyze and save data using *R*.

实践部分最重要的是：

1. 导入数据进入R：因为需要将原始数据集进行预先指定的操作，也就是将导入后的数据集进入函数。
2. 安装与载入包：因为需要包内的function执行操作。
3. 确定好在R内进行数据的分析的底层逻辑，一切都遵循该逻辑进行。
4. 要具有发散思维，这些知识如何被我应用在真实世界？（讲课 和 科学研究）