arXiv:2603.10866v1 [stat.OT] 11 Mar 2026

超越可重复研究：构建数据分析的形式化表示

Roger D. Peng[1]

[1]德克萨斯大学奥斯汀分校统计与数据科学系

摘要

数据分析通常以命令式方式构建，即依次发出代表对数据所采取操作的命令。公开这些命令及数据，对于他人重复分析至关重要。然而，仅呈现代码及运行代码所得的结果，可能会掩盖数据分析人员关于数据的前提、预期和假设等重要细节。理解这种分析推理，对于评估分析质量和提出可能的改进建议至关重要。我们认为，将分析的逻辑构造外显出来的数据分析形式化表示，能够为静态呈现分析人员的推理提供更多有用信息。这种形式化表示能够在无需数据的情况下评估数据分析的某些方面、可视化导向结论的逻辑关联，并评估分析人员的假设对于数据中意外特征的敏感性。本文介绍了这种形式化表示的一种实现，以及如何将其应用于一些常见的数据分析任务。

1 引言

统计学家常常发现自己需要提出不同类型的论断，而这些论断分别由不同类型的证据支持。科学论断由数据支持，这些数据经过公认方法分析，并依据该学科的标准进行解读。例如，在流行病学中，关于疾病病因的论断有时会附带

遵循奥斯汀·布拉德福德·希尔爵士框架的证据（希尔，2015年）。统计学家也常常需要提出描述其工具和方法性质的数学主张。这些主张由另一种类型的证据支持，即逻辑证明。在这两个领域——科学领域和数学领域——都有成熟的方法来呈现证据，以证明某一特定陈述为真。

分析数据的统计学家还会提出第三类主张，我们在此称之为数据分析陈述。数据分析陈述是关于手头数据的陈述，而非关于未观测总体的推断性陈述。“样本的均值为 4.6”或“简单线性回归系数为 1.79”之类的陈述，都是数据分析陈述的例子。这类陈述直接源自正在分析的数据集；与推断性陈述不同，它们可以独立验证其真伪。鉴于数据分析陈述通常是更广泛主张的依据，显然，能够评估这些基础数据分析陈述的有效性至关重要。

在科学文献中，通常认为数据分析陈述无需证明便为真。例如，在医学文献中，结果部分开头通常会列举描述研究人群的汇总统计量，有时还会将这些统计量汇集在一张表格中（Hayes-Larson 等，2019年；Yoshida 和 Bartel，2022年）。读者通常不会得到任何证明或证据来证实这些关于研究人群的陈述为真，而是默认计算这些统计量的人必定知道自己在做什么，并且很可能采用了正确的计算方法。可以说，对于许多数据分析陈述，即使其最终被证明为假，利害关系也不大。但即便要求提供详细证据，目前也不存在一种得到普遍认可的方式来向他人呈现这些证据。

近几十年来，人们对多个领域中计算研究的可重复性日益担忧，因为多项研究被证明其所陈述的结果无法根据原始数据重建（Peng，2011；Peng和Hicks，2021年；Leek和Peng，2015年；国家科学、工程与医学研究院，2019年；Omenn et al.，2012年）。换言之，这些研究中提出的数据分析陈述被证明是错误的。例如，Baggerly及其同事花费了超过1,500小时

试图重现一项重要计算生物学研究的结果（Baggerly和Coombes，2009；Coombes et al.，2007）。他们的调查发现该分析存在大量计算问题，从而得出结论：论文中的所有数据分析陈述都是错误的。诸如此类的高调案例促使许多领域的研究人员要求已发表的论文附带其计算机代码和数据集（Baggerly，2010；Haibe-Kains et al.，2020）。多家期刊已实施政策，以提高其刊载论文的可重复性（Peng，2009；Hofner et al.，2016年；Wrobel et al.，2024年）。

与论文一同发布代码和数据这一主张，内在地将代码和数据视为对论文中数据分析陈述的一种“证明”。然而，单纯提供代码和数据也有一些缺点。首先，作者分享数据和代码的那一刻，实际上并不能证明任何事情。独立分析人员必须在数据上重新运行代码，才能判断分析是否存在问题。根据分析的复杂程度，这可能是一个耗时且耗费资源的过程（Willis 和 Stodden，2021年；Peng，2009），而且还必须手动将计算结果与数据分析陈述关联起来，或将结果放入论文正文中。即使使用文人编程技术创建了组织有序的计算汇编（Gentleman 和 Temple Lang，2007），这一关联过程也并非自动完成。从更广泛的角度来看，分析完成后再提供代码和数据，对于从一开始就防止分析中出现任何问题几乎毫无帮助（Leek 和 Peng，2015年）。可重复性只能用于事后诊断问题，而且还会将验证责任交给独立的第三方，而非原作者。最后，当分析使用敏感数据或受保护健康信息时，可重复性所要求的数据共享可能会带来问题。使用此类数据的出版物很可能永远无法由他人完全复现。

除了上述指出的缺陷之外，可重复性这一概念的一个局限在于，仅凭分析代码通常不足以理解数据分析人员的推理，因此无法评估该推理的质量。关于数据分析的陈述通常不会包含在代码或注释中，而仅为了实现可重复性，也不要求将支持这些陈述的任何证据纳入代码中，以便

实现可重复性。结果，代码说明了为产生某一结果所做的工作，但没有提供任何信息来解释关于数据的任何陈述为何可能为真。

尽管可重复性标准推动了数据分析呈现方式的发展，但归根结底，它是一种动态表示，将数据分析描述为一段需要执行的计算机程序，而非一系列有待核验的陈述或论证。将数据分析描述为计算机程序，可能会导致一个错误的结论：只要程序运行时没有错误，且产生了合理的输出，那么分析就（在某种未明确定义的意义上）是正确的。然而，与其断言这句话通常并不成立，我们不妨问问自己：“能否设计一种用于撰写数据分析的系统，让这句话成立？”我们能否设计一种系统，使计算机程序成功执行就意味着数据分析成功（依据某种定义）？这样的系统可以为数据分析提供一种静态形式化表示（如数学证明），系统地呈现支持某项数据分析陈述的证据链。这种表示还可以弥补上述可重复表示的一些不足，并为交流分析论证和评估数据分析质量开辟新的途径。

1.1 数据分析作为计算机程序

考虑以下可能用于计算 R 中数据框某一列均值的伪代码。

```
dat  <- read.csv("data.csv")
x   <- 数据[, 1]
m   <- 均值(x)
```

从对该代码的静态阅读来看，无法作出任何数据分析陈述，因为作为读者的我们看不到输出。

假设一位分析人员运行了这段代码，结果发现 m 的值是 4.6。随后，分析人员可能会作出如下数据分析陈述：“数据框第一列的均值是 4.6”。在这种情况下，陈述与代码本身是分开的。不过，分析人员也可以改为编写以下代码。

```
dat <- read.csv("data.csv") x <-
dat[, 1]m <- 均值 (x)
stopifnot( round(m, 1) == 4.
6 )
```

从代码的第四个表达式可以看出，我们预期 x 的均值（四舍五入到小数点后一位）为 4.6。因此，我们可以得出结论：如果代码运行时没有出错，那么关于第一列均值的陈述就为真。要得出这一结论，无须查看数据。这个版本的代码通过使用断言，将数据分析陈述以非正式的方式嵌入代码本身。

这个简单的例子清楚地说明了几点。首先，不查看数据或计算结果，就无法编写这样的代码；也就是说，无法预先指定它，因此它不是分析计划。其次，以这种方式编写的代码旨在描述和记录一项分析及其预期行为。第三，这段代码无法在不同的数据集上重复使用；也就是说，它并非通常意义上的计算机程序。这段代码是针对特定数据集开展的数据分析的静态表示，这与数据分析陈述是关于特定数据集的陈述十分相似。最后，这段代码仍使独立研究者能够在需要时重复该分析，因此，以这种方式编写代码并不会丧失可重复性。

在下面几节中，我们将把构建数据分析静态形式化表示的系统形式化，并说明如何在软件中实现它。我们将展示这种表示如何使分析的逻辑构造外显，并介绍一种用于可视化数据分析陈述之逻辑前提的工具。我们还将展示，逻辑构造的分析如何支持对数据分析代码进行静态评估，以及如何在无需对数据运行代码的情况下得出结论。

2 静态形式化表示的原则

在此，我们提出一些一般原则，作为实现数据分析静态形式化表示的基础。任何实现的重要目标，都是在计算分析结果的同时，使数据分析中的逻辑推理外显。

2.1 将数据分析描述为带证据的陈述

回顾第1.1节中的示例，关于数据的原始陈述是：“数据框第一列的均值是 4.6”。对于这个简单的数据分析，只需使用以下代码，就足以声明该分析具有可复现性。

```
compute_mean_1st_col <- 函数(datafile) { dat <-
read.csv(datafile) mean(dat[, 1]) } compute_
mean_1st_col("data.csv")
```

这段代码的结构是：给定一个输入（data.csv 文件），我们会生成一个输出（第一列的均值）。因此，我们可以这样陈述：“compute mean 1st col() 函数以 CSV 文件名为输入，并返回数据框第一列的均值。”通过检查上面的 R 源代码，可以轻松验证这一陈述。然而，这是一条编程陈述¯（而非数据分析陈述），其自然的支持证据是源代码。这段代码所要验证的数据分析陈述（“数据框第一列的均值是 4.6”）尚未得到验证。特别是，即使这段代码能够无错误地执行，如果不检查结果是否与该陈述一致，我们也未必能知道该数据分析陈述是否为真。

将编程范式应用于数据分析问题时，一个关键挑战在于，人们往往会认为数据分析只是接收输入并生成输出。这会导致一种情况：只要能够正确生成指定的输出，分析过程本身就无关紧要。然而，如果我们将数据分析视为生成需要证据支持的数据

分析陈述，那么逻辑构造方法便开始显得合理，因为它要求检查有关数据的前提和假设，以产生支持结论陈述的证据。

2.2 将数据分析陈述视为类定义

我们可以将数据分析陈述视为对某个对象类的描述。因此，“数据框的第一列没有缺失值”这一陈述定义了一个对象类，而创建该类中的一个对象，就表示这一陈述为真。这种形式化方法使我们能够运用熟悉的编程技术来编写有关数据的陈述。

为给定的数据分析陈述关联集中式代码定义，可以提升透明性，并减少对该陈述含义的歧义。定义形式类还可以进行自动有效性检查，并确保调度特定的方法函数。从软件工程的视角来看，采用一个可在多个地方使用的单一定义很有价值，无须每次需要某个概念时都运行临时代码。从统计学视角来看，在数据分析中使用形式定义，可以让我们运用传统逻辑工具证明分析的一些有用性质（第 4 节）。将数据分析陈述视为类定义，最终意味着数据分析中的模块化是围绕我们能够对数据作出的陈述实现的，而不是围绕我们能够应用于数据的操作（即函数）实现的。

2.3 将前提作为类扩展

一旦定义了一个正式类来表示特定的数据分析陈述，就可以扩展该类，将支持这些陈述的前提类纳入其中。使用额外的前提类定义，可以明确规范哪些类型的证据可能支持给定的陈述。将前提嵌入主类定义中的槽，体现了证据层级，而这一层级最终指向原始数据分析陈述。

将支持性陈述关联到结论性陈述的一般方法可以表述如下。设 A 和 B 是两种不同的对象类。“A 支持

B”意味着 B 的类定义中有一个用于存放 A 类对象的槽位。这种方法的优点在于，只要阅读类定义，我们就能知道：只有当 A 是有效对象，且分析人员已决定由 A 类编码的陈述支持由 B 类编码的陈述时，B 对象才会存在。

3 实现

我们将由数据分析产生的数据分析陈述的构建视为分两个阶段进行。首先，必须明确打算就数据作出何种数据分析陈述。其复杂程度可以从对数据的简单汇总，到复杂模型的性能指标。在第二阶段，将数据分析陈述与为其提供支持证据的支持性前提联系起来。一旦有了这些前提，我们可以把每个前提本身视为需要其自身支持性前提的数据分析陈述。因此，前提可能还有其自身的前提，如此等等，并且有可能一直回溯，直到到达分析之外的某个事物，例如外部数据文件，该事物要么被信任，要么以其他方式得到验证。

3.1 数据分析陈述作为类

在第一阶段，即我们指定要作出的数据分析陈述时，可以通过定义一类体现该陈述的对象，在代码中实现这一点。在像 R 这样的编程语言中，我们可以使用 S4 类系统定义一类对象，当且仅当我们想要作出的陈述为真时，这些对象才有效（Chambers，1998）。

3.1.1 示例：缺失数据

请看下面这句话：“数据框的第一列没有缺失值。”虽然这句话不太可能是数据分析的最终结果，但很可能会在更广泛的分析中提及。对于这句话，一个难点在于，人们对“缺失值”的含义并未达成普遍共识。不同学科中的不同惯例可能会导致缺失值编码方式多种多样，

因此，即便只是在此明确说明“缺失值”的形式化定义，也会很有价值。

我们可以在 R 中使用 S4 类编写形式化定义。下面，我们定义一种名为 vector no missing 的对象类，用于描述不含任何缺失值的数值向量。

```
## 无缺失值数值向量的定义 setClass(

    类 = "vector_no_missing", 包含 = "numeric",
    有效性 =函数(object) { if(anyNA(object)) r
    eturn("存在 NA 值") if(any(object == -99))
    return("存在 -99 缺失值") TRUE })
```

vector no missing 类继承自内置数值类。根据其有效性方法，如果数值向量中既没有 NA 值，也没有 -99 值，则将其定义为不含缺失值。根据 S4 类有效性方法的机制，如果在向量中发现 NA 或 -99 值，就会返回错误，并且不会创建该对象。接下来可以运行以下 R 代码，从外部逗号分隔值文件中读取数据。

```
## 字符的强制转换方法 --> 向量_no_missing setAs("character",
"vector_no_missing", function(from) {
    num <- as.numeric(from)
    new("向量_无_缺失", num)
})

## 读取数据并将列强制转换为’向量_no_missing’类
```

```
dat <- read.csv("data.csv", colClasses = "向量_no_missing")
```

从代码本身可以看出，如果没有返回错误，生成的对象 dat 就是一个数据框，其中第一列是一个有效的无缺失值向量类对象，这意味着该列中没有 NA 或 -99 值（事实上，数据框中的所有列都是如此）。因此，我们不需要运行代码并查看结果，就能确定没有缺失值。我们只需要知道代码运行时没有报错。

3.2 识别支持性前提

给定一个经过编码的数据分析陈述，我们希望识别一个或多个前提（即其他数据分析陈述），这些前提能够直接推出结论，或至少为结论提供支持。识别这些前提可以采用直接方法，也可以采用间接方法。采用直接方法时，我们只需找出那些我们知道能够支持结论陈述的数据相关陈述。例如，如果我们想说一个向量的均值是 ≥ 0，那么一个支持性陈述可以是该向量中的所有值都是 ≥ 0。采用间接方法时，我们会考虑结论陈述为真有哪些可能的替代情况？换句话说，我们能否找出该陈述为假的各种条件？如果能，那么我们就可以验证这些条件并未 满足，从而提供证据，说明原结论陈述为真。乍看之下，这种方法似乎有些迂回，但它本质上与我们在一般科学研究中可能采取的方法相似：找出可能的备择假设，利用数据排除这些假设，从而为主要假设提供证据。直接方法和间接方法并无优劣之分，但在某些情况下，其中一种方法可能比另一种更容易实施。

3.2.1 示例：计算列均值

考虑这样一个问题：计算数据框某一列的均值，并提供证据证明它取某个特定值。假设经过一些初步的探索性分析后，我们想要陈述：“数据框第一列的均值是 4。6”。那么，我们可以先定义一类对象：当该陈述为真时，这类对象即为有效对象。下面我们为此定义一个 S4 类。

```
setOldClass("data.frame")
setClass(
    Class = "mean_1st_col_df_is_4.6", slots = c(data = "
    data.frame"), validity =function(object) { first_col <-
    object@data[, 1]if(!isTRUE(all.equal(mean(first_col), 4.6)))
    return("第一列的均值不等于 4.6") TRUE })
```

这个类特别定义了我们所说的“等于”的含义，即 all.equal() 意义上的相等。现在我们可以运行以下代码，来创建我们新定义的对象。

```
dat <- read.csv("data.csv", colClasses = "numeric") x <-
new("mean_1st_col_df_is_4.6", data = dat)
```

根据定义，我们知道 x 是一个对象，它包含一个数据框，其第一列的均值为4.6。

在许多数据分析情形中，我们希望说明的内容不止于手头的基本数据分析陈述，以表明数据中并未出现任何意外情况。例如，如果我们知道数据的分布高度右偏而非对称，那么样本均值为4.6的解释就可能发生变化。在这些情形下，识别支持性前提的间接方法可能会很有用。

我们可以追问：语句“数据框第一列的均值为 4.6”何以可能为假？我们知道，mean() 函数返回的值可能是：(1) 缺失值，(2) 正无穷/负无穷，(3) 小于 4.6，或 (4) 大于 4.6。如果我们聚焦于 mean() 返回 NA 值的情形，那么我们知道，只要能证明第一列没有任何 NA 值，就可以排除 mean() 返回 NA 值的情形。在第 3.1.1 节中，我们已经定义了一个描述无缺失值向量的类。证明数据框的第一列是无缺失值向量类的对象，将

意味着第一列的均值不可能为缺失值。下面的类刻画了语句“数据框的第一列是无缺失值向量类的对象”。

```
setClass(
    类 = "df_1st_col_no_missing", 槽 = c(data = "数据框"),
    有效性 =function(object) { first_col <- object@data[, 1]if(!
    is(first_col, "向量_no_missing")) return("第一列可能包含缺失
    值") validObject(first_col,test = TRUE) })
```

接下来，我们要说明这个对象的存在支持这样一个事实：第一列的均值为 4.6（也就是说，它排除了均值为缺失值的可能性）。为此，我们可以如下定义一个新类。

```
setClass(
    类 = "mean_1st_col_df_is_4.6_WITH_前提", 包含 = "mean_1st_col_df_is
    _4.6", 槽 =c(premise_df_1st_col_no_missing = "df_1st_col_no_missing")
```

这个类通过添加一个槽来扩展现有的 mean 1st col df is 4.6 类，该槽对支持性前提进行编码，即第一列没有缺失值。因此，类定义编码了这样一条陈述：“如果数据框的第一列没有缺失值，那么均值就不可能是缺失值，这也支持了第一列的均值为 4.6 这一陈述。”我们也可以采用类似的方法来说明，如果第一列没有正无穷/负无穷值，那么该列的均值就不可能是正无穷/负无穷（但这里省略这种情况）。

其余两种情况比前两种更为微妙，因为它们可以用几种不同的方式来解释。考虑均值大于4.6的情况。均值可能大于4.6的两种方式是：（a）整个分布向右偏移，或者（b）在右侧存在一些较大的离群值

分布。（对于均值小于4.6的情况，可以采用类似的方法来处理。）有多种统计工具可用于调查并排除这些情况；这里我们将使用简单的五数概括。我们将采用的逻辑是：如果数据的中位数接近4.6（即分布没有向左或向右偏移），并且五数概括未显示显著的离群值，那么这就支持均值为4.6。

首先，我们可以定义两个类，用以刻画两个前提：中位数接近4.6，并且不存在离群值。

```
setClass(
    类 = "中位数_接近_于_4.6", 槽 = c(数据 = "
    数据框"), 有效性 =函数(对象) { first_col <- 对象
    @数据[, 1]m <- 中位数(first_col) if(abs(m -
    4.6) > 0.5) return("中位数不接近4.6") TRUE })
```

```
setClass(
    类 = "五数概括_无_离群值", 槽 = c(数据 = "数据框"),
    有效性 =函数(object) { first_col <- object@data[, 1]f
    n <- 五数概括(first_col) 四分位距 <- fn[4] - fn[2]if(f
    n[5] > fn[3] + 3 *四分位距) return("右侧存在离群值")
    if(fn[1] < fn[3] - 3 * 四分位距 ) return(" 左侧存在离群
    值")
```

```
        TRUE
    })
```

请注意，我们在此明确规定了“接近 4.6”的含义（在 4.6 的 ± 0.5 范围内），以及“显著离群值”的含义（与中位数相差超过或少于 $3 \times IQR$）。

现在，我们必须重写原始类定义：均值为 4.6（数据框第一列），并带有前提，以接受三个论证，因为现在我们有三个前提，而不是一个。只需在原始类定义中添加更多槽即可实现这一点。

```
setClass(
    类 = "均值_1st_col_df_为_4.6_带有_前提", 包含 = "均值_1st_col_df_为_4.6", 槽 = c( 前提_df_1st_col_无_缺失值 = "df_1st_col_无_缺失值", 前提_中位数_接近_到_4.6 = "中位数_接近_到_4.6", 前提_五数概括_无_离群值 = "五数概括_无_离群值" ) )
```

最后，可以运行以下代码执行分析，以表明这些前提支持结论陈述。

```
dat <- read.csv("data.csv", colClasses = "向量_无_缺失值") p1 <- new("df_1st_col_无_缺失值", 数据 = dat) p2 <- new("中位数_接近_到_4.6", 数据 = dat) p3 <- new("五数概括_无_离群值", 数据 = dat) 输出 <-new("均值_1st_col_df_为_4.6_带有_前提", 数据 = dat, 前提_df_1st_col_无_缺失值 = p1, 前提_中位数_接近_到_4.6 = p2, 前提_五数概括_无_离群值 = p3)
```

至此，我们已经排除了开头所指出的四种情形；这些情形都意味着原结论陈述不成立。我们证明了数据

中没有 NA 值，中位数接近 4。6，且右侧（或左侧）没有显著的离群值。这些都是在计算均值时进行数据分析通常会完成的任务。不过，我们将这些活动置于一个逻辑框架中，其中陈述前提以支持结论陈述。

4 代码静态分析

我们提出的数据分析表示方式有一个优点：它使我们能够手动或借助其他程序对数据分析代码进行静态分析。之所以可行，是因为我们的方法要求在代码中明确写出预期、假设和定义。因此，即使不在数据上运行代码，我们也能评估分析的某些方面；如果无法获取数据，这种操作可能代价高昂，甚至根本无法进行。具体而言，创建一个明确定义的类这一要求，迫使分析人员明确说明分析正确意味着什么。此外，我们可以根据代码对分析结果得出一些基本结论，而无需查看输出。

4.1 推断输出属性

再来看第 3.1.1 节中的例子，其中我们有这样一条陈述：“数据框的第一列没有缺失值。”这条陈述由 S4 类“无缺失值向量”（以及相应的强制转换方法）来表征。因此，当我们调用 read.csv()，并使用参数 colClasses = " 无缺失值向量 " 读入数据框时，我们知道，只要代码运行时没有报错，数据框的每一列就都不会有任何缺失值（根据给定的定义）。如果我们在假设数据框至少有一列的前提下运行以下代码，

```
dat <- read.csv("data.csv", colClasses = "向量_无_缺失") 均值(dat
[, 1])
```

我们可以确信，mean() 的输出不是缺失值。因此，无需查看实际结果（这需要运行代码），我们就能推断结果的性质。

4.2 避免静默错误

下面来看一个在实践中很常见、也更微妙的例子：数据框的连接。连接数据框可能会引入隐蔽的错误；如果数据的格式不符合预期，这些错误不会触发任何警告。请看以下两个数据框：

国家 值1 年份 1 美
国 922000 2 美国
1172001 3 美国
932002

and

国家 值2 年份 1 美国
48.74391 2000 2 美国
49.44440 2001 3 美国
50.21478 2002

目标是根据国家列和年份列连接这两个数据框，生成一个包含值1和值2的3行合并数据框。显然，两个数据框中的国家列都旨在表示相同的信息，但第一个数据框使用名称 US，第二个数据框使用名称 USA。这种缩写上的细微差异在实际数据集中很常见，却可能给连接操作带来严重问题。如果假设第一个数据框存储在文件 data1.csv 中，第二个数据框存储在文件 data2.csv 中，那么我们可以设想运行以下代码来执行连接。

```
dat1 <- read_csv("data1.csv") dat2 <- read_csv("data2.csv") merged_f
ull <- full_join(dat1, dat2, by = c("国家", "年份")) merged_left <- left_
join(dat1, dat2, by = c("国家", "年份")) merged_right <-right_j
oin(dat1, dat2, by = c("国家", "年份")) merged_inner <-inner_
join(dat1, dat2, by = c("国家", "年份"))
```

每个连接函数产生的结果都以略有不同的方式出乎意料。由于国家列没有匹配项，而且全连接()会保留原始数据框中的所有行，因此全连接会生成一个包含 6 行的数据框。左连接会生成一个包含 3 行的数据框，但 value2 列全部为缺失值。右连接也会生成一个包含 3 行的数据框，但这次 value1 列全部为缺失值。最后，内连接会生成一个零行的数据框。仅凭阅读代码，若事先不了解数据的样貌，就无法知道这些意外结果中的任何一种会出现。此外，这三种连接都不会产生错误或警告消息。

本示例的主要问题在于，分析人员试图作出的陈述尚未被明确指定。因此，对于正确输出应呈现何种形式，并不存在明确的定义。我们可以使用以下 S4 类，对关于合并数据框的陈述进行编码。

```
setClass(
    Class = "merged_dataset", contains = "data.frame", validity =f
unction(object) { if(nrow(object) != 3) return("行数不正确") if(ncol(object) != 4)
return("列数不正确") if(!all(names(object) %in% c("country", "year", "value1", "
value2"))) return("列名错误") if(anyNA(object$value1) || anyNA(object$value2))
return("'value1' 或 'value2' 中存在 NA 值") TRUE } )
```

该类的有效性方法定义了合并数据框应具备的正确属性，例如行数和列数、列名以及是否存在缺失值。接下来，如果我们运行以下代码，

```
dat1 <-读取_csv("data1.csv") dat
2 <-读取_csv("data2.csv")
merged <-新建("merged_数据集",
                left_join(dat1, dat2, by = c("国家", "年份")))
```

我们知道，它要么会返回一个合并后数据集类的对象，要么有效性方法会返回一个错误（在上面的示例中返回了一个错误）。我们可以根据上面的显式类定义预测这种行为。

4.3 证据与支持的词法分析

由于必须使用 S4 类和方法明确地定义陈述与其支持性前提之间的关系，因此可以通过手动或某种自动化手段来检查代码所允许的这些关系。一般而言，如果定义一个类来表示前提与结论陈述之间的关系，我们就可以检查类定义，以确定哪些确切的证据能够支持该结论陈述。这可以通过直接查看代码中的类定义，或使用 R 中 methods 包的一些工具来完成。

例如，在第 3.2.1 节中，我们定义了一个类 mean 1st col df is 4.6 WITH premise，以允许为陈述“数据框第一列的均值是 4.6”提供前提。如果我们想从代码中了解什么样的证据能够支持这一陈述，就可以直接检查类定义，也可以通过 getClass 函数进行检查。

```
> getClass("均值_1st_col_df_is_4.6_WITH_前提") 类 "均值_1st_col_df_
is_4.6_WITH_前提" [位于 ".GlobalEnv"]

槽:

名称: 前提_df_1st_col_no_missing 前提_中位数_接近__4.6
类: df_1st_col_no_missing 中位数_接近__4.6

名称: 前提_五数概括_无_离群值 data
```

类:　　五数概括_无_离群值　　数据框

扩展："均值_1第_列_数据框_为_4.6"

从 getClass 列表中可以看到，一个这样的支撑性前提类是中位数接近 4.6 类。想了解这个类定义的人可以直接查阅，尤其是它的有效性方法。

```
> getClass("中位数_接近__4.6")@有效性 函数
(object) { 第_列 <- 对象@数据[, 1]m <- 中位数(第_
列) if(abs(m - 4.6) > 0.5) return("中位数不接近
4.6") TRUE }
```

当然，在 R 中运行 getClass 函数需要在 R 中执行部分分析代码。不过，在这种情况下，我们只需执行类定义代码即可。我们不需要从类本身生成任何对象，因为这需要访问数据。

4.4 前提结构可视化

如果前提不止几个，getClass 的输出可能会很繁杂，但类的层次结构天然适合用基于树的可视化方式呈现。我们编写了一些代码，用文本形式帮助可视化前提层级（代码清单见补充信息附录 A）。

```
> 陈述("均值_1st_col_df_is_4.6_WITH_前提")

                层级名称
1 根目录
2 °--均值_1st_col_df_is_4.6_WITH_前提 3 ¦--df
_1st_col_no_missing
```

4 ¦--中位数_接近_于_4.6 5 °--五数
概括_无_离群值

我们还编写了相应的绘图方法，将支持性前提的树结构可视化为图形（图1）。

无论是在文本输出中还是在图中，解读方式都是：通过同一个父节点连接的子节点，彼此之间以逻辑与绑定。因此，子节点中编码的所有前提都必须成立，父节点才成立。在某些情况下，查看前提层级的可视化就足以评估数据分析或识别潜在弱点，无需检查代码。

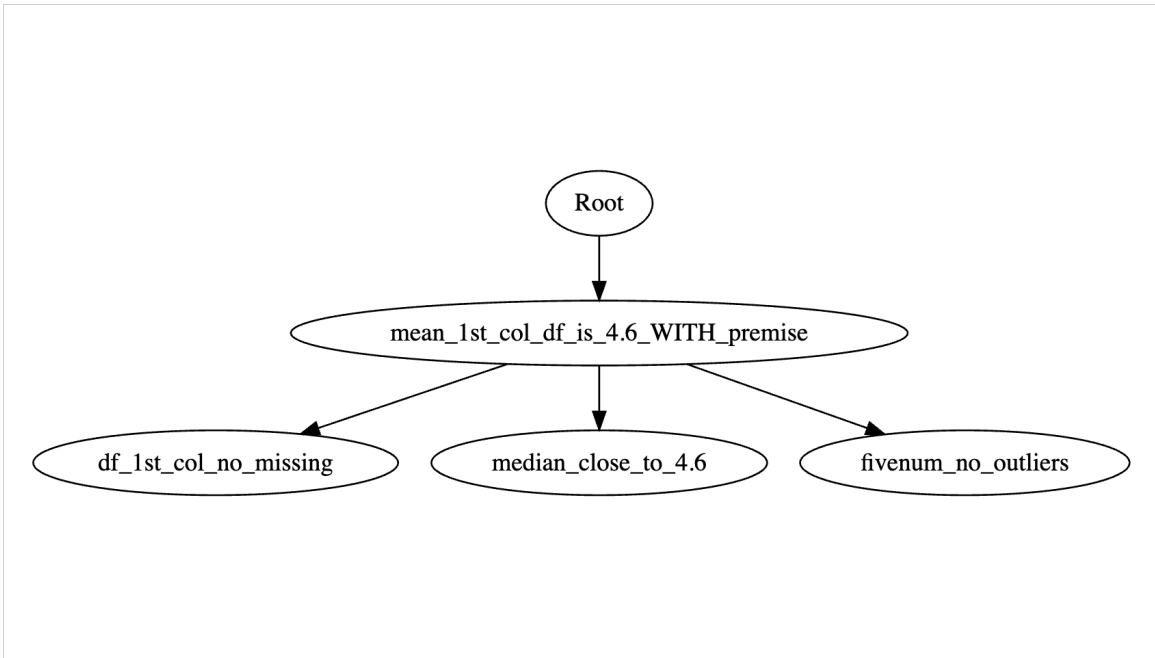


图1：用于估计均值的支持性前提层级。

4.5 敏感性分析

将数据分析陈述实现为 S4 类的一个关键优势是，每条陈述都通过有效性方法与一种检验其有效性的方法配对。这样，我们便可以利用这些内置测试，检验结论对数据中意外特征的敏感性。

这些意外特征，例如离群值或非线性，可以通过模拟生成，并在这些替代数据集上运行代码。Zhang 和 Peng（2026年）描述了这种方法的一个版本。这里的一个关键目标可能是模拟具有意外特征、但不会被前提陈述中的各种有效性方法“捕获”的数据。在这种情况下，可能需要添加更多前提陈述，或调整现有前提陈述中有效性方法的容差。

请看第 3.2.1 节中计算向量均值的示例。我们可以生成均值为 4。6 的数据（因此结论陈述为真），但这些数据或许还具有一些出乎意料的特性，例如存在较大的离群值或呈偏态分布。模拟此类数据并评估前提陈述的有效性，有助于确保在得出结论陈述时，数据确实是以预期方式生成的。

5 应用：简单线性回归

本节将通过一个较长的示例，说明如何将我们的方法应用于涉及简单线性回归的问题。在许多线性回归应用中，研究者会报告一个关注的系数，该系数可能表示某种关注的暴露与某个结局之间的关联。与此同时，仅仅陈述系数（及其不确定性）通常并不足以涵盖分析人员所要表达的全部内容。分析人员还会希望证明数据中没有发生异常或意外情况，并且我们对数据生成过程有一定合理的了解。要作出这样一个综合性陈述，通常需要对数据进行多项检查，尤其是检查模型残差。

为演示之用，我们模拟了一些二元数据，其简单线性回归斜率为 1.79。在对该数据集进行分析时，我们可能想要表达这样的陈述：“回归线的斜率为 1.79”。这样的陈述可以编码为以下类。

```
setClass(
    类  =  "slr_斜率_1.79",
```

槽 = c(拟合 = "线性模型"), 有效性 =函数(object) { 贝塔 <-
round(coef(object@fit), 2) if(贝塔[2] != 1.79) return("斜率系数与
主张不匹配") TRUE })

除了这条主要陈述，我们还希望提供支持性前提。这些前提将涉及以下陈述：(1) 残差不存在非线性；(2) 响应变量或预测变量中不存在显著离群值；以及 (3) 残差图是否恰当。我们可以通过以下方式扩展上面的类，加入这些前提。

setClass(
类 = "slr_slope_1.79_WITH_前提", 包含 = "slr_slope_1.79", 槽
= c(前提_slr_无_非线性 = "slr_无_非线性", 前提_slr_无_离群值 = "
slr_无_离群值", 前提_slr_图_看起来_合适 = "slr_图_看起来_合适"))

类定义中指出的三个前提类仍需定义，我们在补充信息附录B中展示了完整的实现。要检查非线性，我们可以定义以下类，它只检查数据中是否存在可能的二次结构。

setClass(

Class = "slr_no_nonlinearity", slots = c(fit = "lm"), validity = function(object) { fit_alt <- lm(y ~poly(x, degree = 2), data = model.frame(object@fit)) llr <- as.numeric(logLik(fit_alt) - logLik(object@fit)) if(llr >= 7) return("数据中存在强烈的非线性证据") TRUE })

在这里，如果二次模型与原始线性模型之间的对数似然比大于 7（即二次结构的证据非常有力），有效性方法就会返回一个错误。

其中一个前提涉及通过两个标准残差图检查残差：标准化残差的直方图，以及拟合值与残差的散点图。

setClass(Class = "slr_plots_look_okay", 槽 = c(前提_残差_直方图_正常 = "残差_直方图_正常", 前提_拟合值_与_残差_正常 = "拟合值_与_残差_正常"))

同样，此处仍需定义这两个前提类。下面展示其中一种。鉴于自动检查可视化图形并非易事，我们在此采用人工方式，要求分析人员查看图形并交互式确认其是否

符合预期（具体标准由分析人员自行决定）。下面展示如何实现对标准化残差直方图的检查。分析人员确认直方图可接受后，有效性方法便会返回；否则，将返回一个错误。

setClass(

类 = "残差_直方图_是否正常", 槽 = c(fit = "线性模型"), 有效性 =函数 (object) { r <- local({ res <- rstandard(fit) hist(res, main ="标准化残差") readline("直方图看起来正常吗？ [y/n] ") }) if(tolower(r) !="y") return("标准化残差直方图有问题") TRUE })

完整的前提树（图 2）可以使用 Supporting Information AppendixA 中实现的绘图方法进行可视化。

与我们的方法相比，简单线性回归问题的传统分析计划可能如下所示。

拟合模型并检查系数 fit <- lm(y ~ x, dat) coef(fit)

诊断图

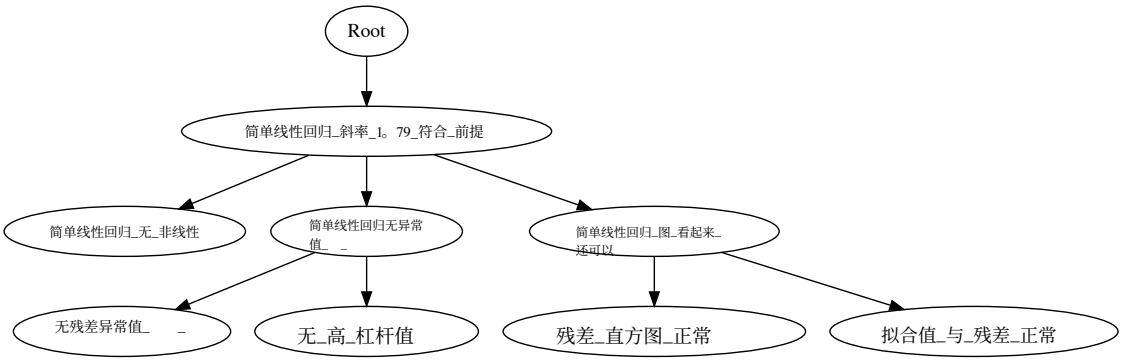


图 2：简单线性回归陈述前提的层次结构。

```
hist(rstandard(fit))
plot(fitted(fit), resid(fit))


## 检查杠杆值和残差摘要
(summary(hatvalues(fit))
summary(rstandard(fit)))


## 检查二次非线性 fit_alt <- lm(y ~ poly(x,
degree = 2), dat3) logLik(fit_alt)- logLik(fit)
```

虽然方法的执行与我们的方法大体相同，但这份代码清单缺少对分析人员预期的任何理解。仅仅知道上述代码执行时没有出错，并不能让我们了解多少有关分析的信息，至多只能说明采用了常见的工具。

6 讨论

我们提出了一种呈现数据分析的新方法，以透明的方式将分析人员对数据的假设和预期外化，使他人能够对其进行评估、解读和批评。我们将数据分析表示为由数据分析陈述构成的逻辑构造，这些陈述以层次化方式相互支持。

我们以可重复研究背后的理念为基础（Peng 等，2006年；Peng，2011），在这种研究中，代码仅用于复现输出；同时，我们在代码之上增加了语义层，使独立分析人员能够理解代码所表达意图的含义。我们将数据分析陈述实现为类定义，其中前提类表示不同数据分析陈述之间的关系。如示例所示，这种传达数据分析的方法的一个关键优势在于，它迫使分析人员明确预期结果是什么。这种明确的表示方式使得人们能够对代码以及分析本身的质量进行静态分析。与可重复分析不同，逻辑构造的分析并不一定要求在数据上执行代码，才能理解该分析的重要特征。

这种数据分析表示法的某些方面借鉴了计算机科学中的命题即类型原则（Harper，2016）。在该原则中，命题被视为编程语言中的类型，而该类型的项的存在则被视为命题为真的证明。在这里，我们使用 R 中的类系统来刻画数据分析陈述，并将属于该类的对象的存在作为该陈述为真的指示。此外，我们还加入了一个用于构建和刻画支持性前提的系统；如果要利用数据分析得出更有力的结论，这一点非常重要。与命题即类型原则类似，我们将表示数据分析陈述的类视为编程语言中的常规对象。这些对象可以像其他对象一样进行操作，因此，它们所编码的陈述可以重新用于支持其他数据分析。

这里所述的工作也与单元测试这一概念有一定关联（Zhu 等，1997；Wickham 和 Bryan，2023；Zhang 和 Peng，2026）。一般而言，单元测试可用于确保编写的软件或函数符合某项规范，方法是检查特定输入是否与预期输出相符。单元测试要发挥作用，关键在于清楚了解给定函数的预期输出是什么。与我们通过逻辑构建的数据分析类似，单元测试所要求的明确规范，使其成为软件开发中的有效工具。不过，单元测试的应用背景与我们在这篇论文中所描述的不同。我们并非

要确保某个软件包符合既定规范，而是旨在通过一系列相互关联的数据分析陈述来表述数据分析论证。

在我们的框架中，一个明确的问题是，如何才能最好地纳入图形摘要和方法。一般而言，在绘制图形之前，很难确定如何表达对图形摘要的预期。已有一些相关工作致力于促进图形比较，例如使用栅格图像和逐像素比较进行单元测试（Murrell，2023年）。图像的文本格式（如可缩放矢量图形（SVG）格式）使得可以应用文本差异工具来评估变化（Henry 等人，2024年）。然而，这些方法以及图像哈希等相关方法（Swaminathan 等人，2006年），都要求先生成图像，才能以其为基础进行比较。将评估可视化摘要是否符合预期这一通常凭直觉进行的过程形式化并编写成代码，很可能颇具挑战，也有待进一步研究。

我们用于可视化数据分析前提的层次树，与系统工程领域之间存在一种有趣的联系。图 1 和图 2 中的树与故障树这一概念相关。故障树是系统工程中常用的一种工具，用于开展结构化风险评估，在航空、航空航天和核电领域的应用历史悠久（Vesely 等人，1981年；Michael 等人，2002年）。故障树本质上是图 1 和图 2 中所示树的逻辑否定。我们的树表示，要使顶层陈述为真（即成功），哪些条件必须成立；而故障树则表示，要使意外结局发生（即失败），哪些条件必须成立。从某种意义上说，故障树与我们在第 3.2 节中描述的用于识别支持性前提的间接方法有关。故障树在系统工程文献中已有充分应用，然而我们很难在统计分析文献中找到对这一概念的明确应用。在构建数据分析的逻辑结构时，故障树或许有助于识别数据分析陈述的支持性前提，因为它能提供可通过数据检验的情景。

这里使用 R 中的 S4 类和方法系统，为这种逻辑构造如何得以实现提供具体示例。显然，这种特定的实现过于冗长，不适合实际的数据分析。为每条

语句创建类会产生大量代码开销。不过，值得注意的是，其中许多额外代码都用于明确规定我们期望在输出中观察到的定义。虽然简单的分析可能只需几行代码即可实现可复现，但从这里的示例可以看出，这样的呈现方式省略了大量信息。无论使用 S4 类还是其他方法，要明确呈现这些信息，都需要更多代码。话虽如此，很可能可以设计出一种更高效的系统来传达这些信息，而将这样的系统添加到 R 中，或许会成为未来颇有价值的研究方向。

归根结底，我们所提出的是对数据分析中已完成工作的形式化表示，而非用于进行数据分析的工具。因此，我们并不认为这种表示本身就能提高一项分析的质量。然而，若要普遍提高数据分析质量，一个有用的工具便是清晰呈现分析人员的思考与推理。我们相信，与当前侧重于可重复性的方法相比，我们的框架能够使这种推理更加透明。

参考文献

Baggerly, K.（2010），“在出版物中披露全部数据”，《自然》，467，401–401。

Baggerly, K. A. 与 Coombes, K. R.（2009），“从细胞系推导化学敏感性：高通量生物学中的法证生物信息学与可重复研究”，《应用统计学年刊》，1309–1334。

Chambers, J. M.（1998），《数据编程：*S* 语言指南》，施普林格。Coombes, K.、Wang, J. 与 Baggerly, K.（2007），“微阵列：回溯步骤”，《自然医学》，13，1276–1277。

Gentleman, R. 和 Temple Lang, D.（2007），《统计分析与可重复研究》，《计算与图形统计学杂志》，16，1–23。

Haibe-Kains, B., Adam, G.A., Hosny, A., Khodakarami, F., 大规模分析质量控制（MAQC）学会董事会, Waldron, L., Wang, B., McIntosh, C.,

Goldenberg, A., Kundaje, A., Greene, C.S., Broderick, T., Hoffman, M.M., Leek, J.T., Korthauer, K., Huber, W., Brazma, A., Pineau, J., Tibshirani, R., Hastie, T., Ioanni- dis, John P. A. Quackenbush, J., and Aerts, H. J. W. L. (2020), “人工智能的透明性与可重复性”，《自然》，586，E14–E16。

Harper, R. (2016年),《编程语言的实用基础》, 剑桥大学出版社。

Hayes-Larson, E., Kezios, K. L., Mooney, S. J., and Lovasi, G. (2019年), “这项研究中究竟有哪些人？制定实用表1的指南”，《临床流行病学杂志》, 114, 125–132。

Henry, L., Pedersen, T. L., Luciani, T. J., Decorde, M., and Lise, V. (2024年), *vdiffr*：视觉回归测试与图形差异比较, R 包版本 1.0.8。

Hill, A. B. (2015年), “环境与疾病：关联还是因果关系？”《英国皇家医学会杂志》, 108, 32–37。

Hofner, B., Schmid, M., and Edler, L. (2016年), “统计学中的可重复研究：《生物统计学杂志》综述与指南”，《生物统计学杂志》, 58, 416–427。

Leek, J. T. 和 Peng, R. D.（2015年），“观点：可重复研究仍可能出错：采用预防方法”，《美国国家科学院院刊》，112，1645–1646。

Michael, S.、Joane, D.、Joseph, F.、Joseph, M. 和 Jan, R.（2002），《航空航天应用故障树手册》，*NASA*安全与任务保证办公室-*NASA*总部，华盛顿，2–8。Murrell（2023），*gdiff*：图形差异测试，R 包版本 0.2-5。国家科学、工程与医学研究院（2019），科学中的可重复性与可重复研究，国家科学院出版社。Omenn, G. S.、Nass, S. J. 和 Micheel, C. M.（2012），转化组学的发展：经验教训与未来方向，国家科学院出版社。

Peng, R. D.（2009），《可重复研究与生物统计学》，生物统计学，10，405–408。—（2011），《计算科学中的可重复研究》，《科学》，334，1226–1227。Peng, R. D.、Dominici, F. 和 Zeger, S. L.（2006），《可重复的流行病学研究》，《美国流行病学杂志》，163，783–789。Peng, R. D. 和 Hicks, S. C.（2021），《可重复研究：回顾》，《公共卫生年度评论》，42，79–93。Swaminathan, A.、Mao, Y. 和 Wu, M.（2006），《鲁棒且安全的图像哈希》，《*IEEE*信息取证与安全汇刊》，1，215–230。Vesely, W. E.、Goldberg, F. F.、Roberts, N. H. 和 Haasl, D. F.（1981），《故障树手册》，技术报告，美国核管理委员会，华盛顿特区。Wickham, H. 和 Bryan, J.（2023），*R* 包，"O'Reilly Media, Inc."。

Willis, C. 和 Stodden, V.（2021年），"信任但要验证：如何利用政策、工作流程和基础设施确保出版物中的计算可重复性"，《哈佛数据科学评论》，2。

Wrobel, J., Hector, E. C., Crawford, L., McGowan, L. D., da Silva, N., Goldsmith, J., Hicks, S., Kane, M., Lee, Y., Mayrink, V., 等（2024年），《与作者合作提升 JASA 的可重复性》，《美国统计学会杂志》，119，795–797。Yoshida, K. 和 Bartel, A.（2022年），*tableone*：创建*"表1"*，用于描述基线特征（含或不含倾向评分权重），R 包版本 0.13.2。Zhang, H. S. 和 Peng, R. D.（2026年），《由内而外：将数据分析中的假设外化为验证检查》，《数据科学杂志》，即将发表。Zhu, H.、Hall, P. A. 和 May, J. H.（1997年），《软件单元测试覆盖率与充分性》，《*ACM*计算调查》，29，366–427。

补充信息

可视化支持性前提

setClass(Class = "陈述_对象", 槽
= c(输出 = "列表"))

根据类名查找前提 get_前提 <-函数(clname) { def <-
getClass(clname) s <- slotNames(def) m <- grepl("^前提", s) if(!
any(m)) { return(list(NULL)) } else { 前提 <- lapply(s[m], 函数(
nm) def@slots[[nm]]) names(前提) <- unlist(前提) lapply(前提, get_
前提) } }

将前提整理成层次化列表 statements <- 函数(x) { if(isS4(x)) cl
<- 类(x) else if(is.character(x)) cl <- x else stop("无法处理对象")
pr <- get_ 前提 (cl) 输出 <- 列表 (pr) names(输出) <- cl new("
陈述_对象", 输出 = 输出) }

打印前提陈述的树结构 setMethod("show", "
statement_object", function(object) {

```
        tree <- data.tree::as.Node(object@输出)
    print(tree) invisible(tree)

)

## 绘制前提陈述集的树结构 setMethod( "plot", "
statement_object", function(x, y, ...) { tree <-
data.tree::as.Node(x@output) plot(tree, ...) } )
```

B 简单线性回归示例类

```
setClass( 类 = "slr_slope_1.79", 槽 = c( 拟合 = "线性模型" ), 有效
性 =function(object) { 贝塔 <- round(coef(object@fit), 2) if(贝塔
[2] != 1.79) return("斜率系数 does not match 主张") TRUE } )
```

```
setClass( Class = "slr_no_nonlinearity", slots = c( fit = "lm" ),
validity =function(object) { fit_alt <- lm(y ~poly(x, degree = 2), data =
model.frame(object@fit)) llr <- as.numeric(logLik(fit_alt) -
logLik(object@fit)) if(llr >= 7) return("数据中存在强烈的非线性证据")
TRUE } )
```

```
setClass( 类 = "slr_no_离群值", 槽 = c( 前提_no_resid_离群值
 = "no_resid_离群值", 前提_no_high_leverage = "no_high_
leverage" ) )
```

```
setClass( 类 = "无_残差_离群值", 槽 = c( fit = "线性模型" ), 有效性 =函数(对象)
{ res <- rstandard(对象@fit) if(any(abs(res) > 4)) return("某些标准化残差大于 +/-
4") TRUE } ) setClass( 类 = "无_高_杠杆", 槽 = c( fit = "线性模型" ), 有效性 =函
数(对象) { h <- hatvalues(对象@fit) if(any(h > 5 * 均值(h))) return("某些点具
有非常高杠杆") TRUE } ) setClass( 类 = "简单线性回归_图_看起来_正常", 槽 =
c( 前提_残差_直方图_正常 = "残差_直方图_正常", 前提_拟合值_对_残差_正常 = "
拟合值_对_残差_正常" ) ) setClass( 类 = "残差_直方图_正常",
```

槽 = c(fit = "线性模型"), 有效性 =函数(对象) { r <- local({ res <- rstandard(fit) hist(res, main ="标准化残差") readline("直方图看起来正常吗？ [y/n]") }) if(tolower(r) != "y") return("标准化残差直方图存在问题") TRUE })

setClass(Class = "拟合_与_残差_看起来正常", slots = c(拟合值 = "线性模型"), 有效性 =函数(object) { r <- local({ g <- fit |>augment() |> ggplot(aes(.fitted, .resid)) +geom_point() +geom_hline(yintercept = 0, lty = 2) +labs(x =" 拟合值 ", y = " 残差 ") print(g) readline(" 这个残差图看起来正常吗？ [是/否] ") }) if(tolower(r) !="是") return("拟合值与残差图存在问题") TRUE })

setClass(Class = "slr_斜率_1.79_符合_前提", contains = "slr_斜率_1.79", slots = c(前提_slr_不存在_非线性 = "slr_不存在_非线性", 前提_slr_不存在_离群值 = "slr_不存在_离群值", 前提_slr_图_看起来_正常 = "slr_图_看起来_正常")

)