TAPPS 第 1 版：面向技术分析与应用统计的可扩展插件的平台

Justin Sam Chew

Colossus Technologies LLP，新加坡共和国
chewjustinsam@gmail.com

Maurice HT Ling

Colossus Technologies LLP，新加坡共和国 生物科学学院，墨尔本大学 澳大利亚维多利亚州帕克维尔 3010 mauriceling@colossus-tech.com

摘要

我们介绍 TAPPS（技术分析与应用统计系统）的首个版本。它是一个以技术分析和应用统计为目标、基于 Python 实现的轻量软件平台。TAPPS 的核心是用于容纳二维数据框对象的容器，以及 TAPPS 命令语言。TAPPS 语言并非用于脚本编写和插件开发的编程语言，而是面向实际操作。就这一点而言，TAPPS 语言更接近 SQL 而非 R，因此学习曲线更平缓。所有分析函数均以插件形式实现。这形成了一套明确的插件系统，可支持分析函数的快速开发与集成。TAPPS 第 1 版根据 GNU 通用公共许可证第 3 版发布，可用于学术和非商业用途。TAPPS 代码仓库位于 http://github.com/mauriceling/tapps。

关键词：应用统计、技术分析、可扩展插件、平台、Python。

1. 引言

如今市场上有许多统计平台，可从多个不同角度进行分类。按许可证类型划分，有些平台采用专有许可（如 SAS [1] 和 Minitab [2]），另一些则采用开源许可（如 SOCR [3]和 ELKI [4]）。按命令行界面（CLI）和图形用户界面（GUI）划分，有些平台基于图形用户界面（如 SOFA [5] ）或命令行界面（如 ASReml [6] ），另一些平台（如 ELKI [4] 和 JMP [7]）则同时提供图形用户界面和命令行界面。在基于命令行界面的平台中，有些平台（如 SciPy [8]）是已知编程语言的库，或开发了自己的图灵完备编程语言（如 R [9]）；另一些平台（如 TSP [10]）则实现了一种领域专用命令语言，其目标并非成为图灵完备语言。领域专用命令语言的主要优势在于，与完整编程语言相比，学习曲线更平缓；SQL 就是一个广受欢迎的例子。

本文介绍 TAPPS 第 1 版（“技术分析与应用统计系统”的缩写，以下简称“TAPPS”），这是一个以技术分析和应用统计为目标的平台。TAPPS 根据 GNU 通用公共许可证第 3 版授权，可用于学术和非营利用途。TAPPS 的主要特点包括：（1）轻量平台；（2）基于命令行界面的领域专用命令语言；（3）所有分析函数均以插件形式实现。这形成了一套明确的插件系统，可支持分析函数的快速原型开发与测试。TAPPS 代码仓库位于 http://github.com/mauriceling/ tapps。

2. 架构与实现

TAPPS 采用 Python编程语言实现，得益于其固有的语言特性 [11],，这些特性有助于提高代码可维护性。截至本文撰写时，作者之一同时也是 COPADS（ https://github.com/mauriceling /copads）的项目架构师和主要开发者。COPADS 是一个算法与数据结构库，完全使用 Python编程语言开发，且没有第三方依赖。

本节将详细介绍 TAPPS 的架构与实现，细致程度足以让感兴趣的开发者分叉该代码并进一步改进。

TAPPS 由 4 个主要子系统组成（图 1），并由第三方库提供支持。用户通过

其命令行界面与 TAPPS 交互。指令使用 Python-Lex-Yacc [12] 解析为字节码指令。TAPPS 虚拟机执行这些字节码，将数据从多数据框对象（数据持有器）传输到一个或多个插件（功能），并将插件的结果加载回多数据框。

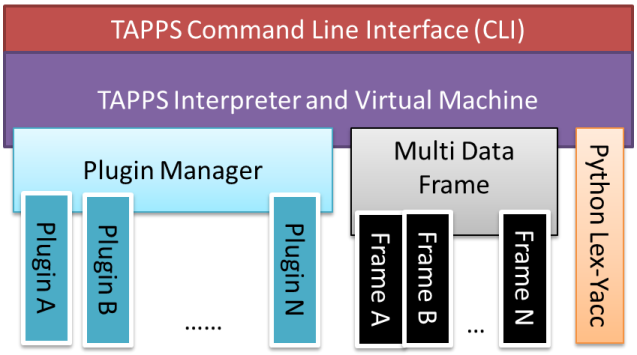

图1 TAPPS 架构。

多数据框。TAPPS 使用 COPADS 中的数据框对象（文件：dataframe.py）作为其主要数据结构（图1）；该对象本质上是一个二维数据表。在实际使用中，一个会话通常包含多个数据框对象。例如，用户可以将一个或多个 CSV 文件加载到 TAPPS 中，其中每个 CSV 文件都作为一个数据框进行维护。此外，用户还可以根据值将一个数据框切分为多个数据框，或复制数据框。因此，为了容纳多个数据框，TAPPS 使用 COPADS 的多数据框对象，而不是直接使用 COPADS 的数据框对象，因为多数据框对象是一个用于容纳一个或多个数据框对象的容器。

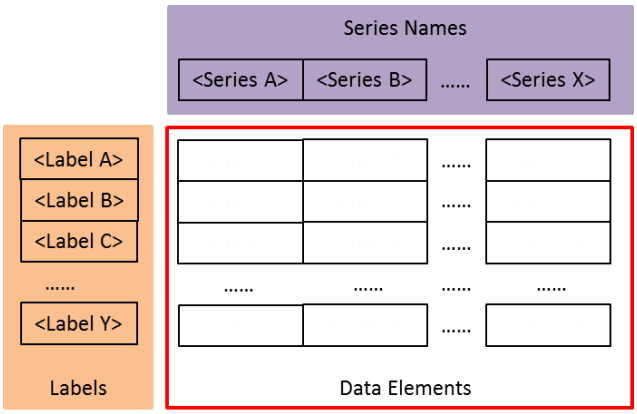

图2 数据框示意图。

数据框的核心（图 2）是一个 Python 字典，其中每个数据行都是一个带有唯一键的值列表。这意味着每个数据行都有一个行标签。同一数据框中的每个数据行必须包含相同数量的数据元素。行中的每个数据元素或值都归属于一个序列名称，这意味着每个序列以表格形式表示时都是一列。这确保每个数据框都是矩形或正方形。

插件和插件管理器。作为一个轻量级平台，TAPPS 不包含任何分析函数，本身也不执行任何操作。所有分析函数（包括统计检验程序）都通过插件提供。因此，TAPPS 的功能丰富程度完全取决于插件的种类。每个插件都实现为一个独立Python模块，其形式为一个至少包含3个文件的文件夹——（1）初始化文件（文件：__init__.py），使Python虚拟机能够将该文件夹识别为可导入模块；（2）清单文件（文件：manifest.py），其中包含用于描述插件的属性；以及（3）操作文件（文件：main.py），其中必须包含一个“main”函数，作为插件的执行入口点。此外，“main”函数以单个“参数”字典作为输入参数，并将修改后的“参数”字典返回给 TAPPS。所有插件都必须位于 TAPPS 中的“plugins”文件夹内（有关编写插件的简要说明，请参阅附录B）。启动时，插件管理器会扫描“plugins”文件夹内的所有独立子文件夹，将其作为潜在插件，并尝试加载每个潜在插件。

“参数”字典构成了 TAPPS 与其插件之间的通信链接；因此，必须在 main.py 文件中为特定插件包含一个空的“参数”字典。这就好比客户（TAPPS）将自己的笔记本电脑连同指令（“参数”字典）送到电脑硬件店（插件）。硬件店的服务人员（插件的“main”函数）打开笔记本电脑，根据“参数”字典中的选项执行所需的硬件和软件升级或服务，然后合上电脑，将笔记本电脑连同服务收据／报告一并交还给客户（TAPPS），作为最初服务指令的凭证。因此，“参数”字典由三个主要部分组成——输入数据框（送修前的笔记本电脑）、结果数据框（维修后的笔记本电脑及服务收据／报告），以及一组选项；这些选项可能因插件而异（服务要求）。

命令行界面。TAPPS 命令行界面（CLI）用于促进 TAPPS 与用户之间的交互。CLI 可通过两种不同模式启动：交互模式或脚本模式。在任一模式下，都必须等前一条命令执行完毕后才能执行下一条命令（有关 TAPPS 语言的定义，请参见附录C）。在交互模式下，TAPPS 的行为类似于 SAS [1] 和 R [9] 命令行界面，用户

依次输入每条命令，并按输入顺序执行。

在脚本模式（也通常称为批处理模式）下，用户将以 TAPPS 语言编写的纯文本脚本提供给 TAPPS，该脚本用于表示按顺序执行的一批命令。这样一来，实现计算密集型统计算法 [13]，（例如蒙特卡罗方法）[14], 的插件便可运行任务，而无需用户干预。脚本的一个重要特性是可以包含一个或多个其他脚本（图 3），从而实现脚本的重复利用并提高其可维护性。这是通过“include”预处理器“@include”实现的，其语法与 C 语言中的“#include”完全相同。在运行机制上，“include”预处理器表示另一个脚本文件的替换位置。因此，只要不存在循环包含，就可以递归包含脚本文件。

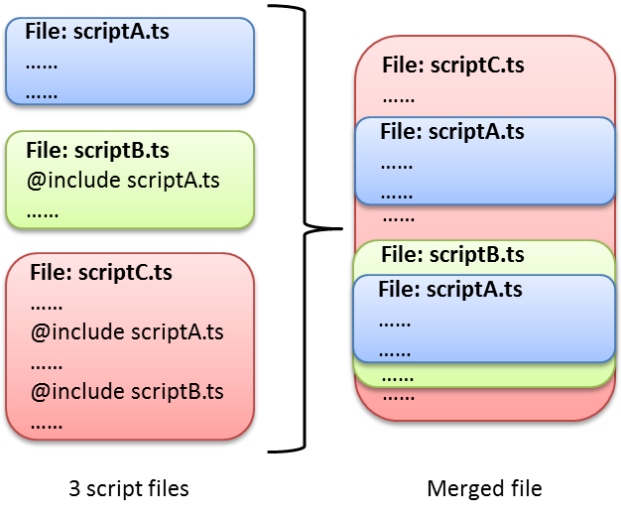

图 3 脚本文件的导入与合并。

解释器和虚拟机。通过 CLI 输入 TAPPS 的所有命令，无论处于交互模式还是脚本模式，都会由 TAPPS 解释器（由 TAPPS 词法分析器和语法分析器组成，并使用 Python-Lex-Yacc [12] ）处理，转换为 TAPPS 字节码（有关 TAPPS 字节码的定义，请参见附录 D）。TAPPS 字节码遵循事实上的格式 [15] ：以一条指令（通常称为操作码）开头，后跟零个或多个操作数。

生成的字节码将作为 TAPPS 虚拟机（TVM）的输入。TAPPS 虚拟机由 3 个主要组件组成：(1) 一组以 Python 字典实现的环境变量，用于保存所有环境设置；(2) 一组以 Python 字典实现的会话变量，用于保存所有数据对象；以及 (3) TAPPS 字节码执行器。

以 Python 字典实现的会话变量集包含 3 个主要变量。首先是多数据框对象（session['MDF']），用于保存所有数据框；这些数据框可在 session['MDF'].frames 中找到。例如，标识符为“DataA”的数据框可通过 session['MDF'].frames['DataA']访问。其次，所有插件参数字典都存储在“parameters”（session['parameters']）中；它以 Python 字典实现，并以参数的标识符作为字典键。例如，标识符为“pluginA_test”的参数集/字典可通过 session['parameters']['pluginA_test']访问。最后，所有潜在插件都会记录在 session['plugins']，中，该变量以 Python 字典实现。成功加载的插件列表将列于 session['plugins']['loaded'] 中，而加载失败的插件列表将列于 session['plugins']['loadFail']中。

对于每个成功加载的插件，系统都会在会话中创建一个新字典，以“plugin <plugin name>”为键，用于存储与该插件相关的信息和参数字典。例如，如果成功加载了“ztest”插件，则可通过 session['plugin ztest']访问有关“ztest”插件的信息。同时，session['plugin_ztest'] ['main'] 将指向 ztest 的主函数，而 ztest 的参数字典可通过 session['plugin ztest']['parameters']查找并复制。

TAPPS 字节码执行器采用大型 IF 语句实现，根据操作码将字节码操作数（如果有）传递给相应的辅助执行器。每个辅助执行器随后接收给定的操作数，并对会话变量和环境变量进行操作。

3. TAPPS 语言

TAPPS 语言定义了 13 种语句（附录C），可分为两部分——元命令和操作指令。元命令语句主要用于设置变量和执行显示命令，以管理当前会话的行为或提供相关信息。以下介绍 4 种元命令语句：

- 描述：描述数据对象或变量，比“显示”语句提供的信息更多

- 设置：设置环境中的选项或插件参数字典中的选项
- Shell：启动语言子 Shell，或执行非 TAPPS 语句
- 显示：显示会话或环境变量/项目的值。

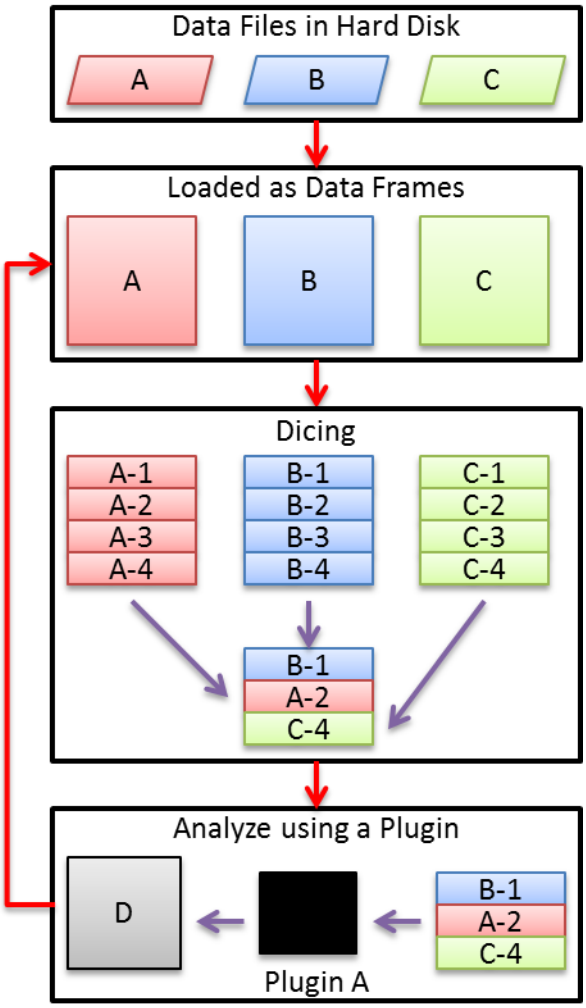

图 4 加载-切分-分析循环。将三个单独的数据文件（A、B 和 C）分别加载到独立的数据框（A、B 和 C）中。每个数据框再切分为 4 个较小的数据框；例如，数据框 A 被切分为数据框 A-1、A-2、A-3 和 A-4。将每个原始文件中的一个数据框（B-1、A-2 和 C-4）合并为一个数据框，然后由插件 A 对其进行分析。插件 A 的输出为数据框 D，可将其附加为下一轮循环的数据框。

操作语句负责管理 TAPPS 的功能和操作；更具体地说，是指加载-切分-分析循环（图 4）。在此循环中，首先通过读取一个或多个数据文件将数据加载到 TAPPS 中。下一步是切分，即提取已加载的数据和/或将数据片段合并在一起。例如，将多个已加载的 CSV 文件导入数据框后，可以根据选择条件将其切分为多个较小的数据框，然后按逻辑将切分后的数据框合并为一个较大的数据框。最后，可以使用插件对合并后的数据框进行技术分析和/或统计分析。插件分析的输出结果是一个数据框对象，因此可以将其附加到多数据框对象中，以便重复加载-切分-分析循环。

以下介绍 9 条操作语句：

- 类型转换：将值转换为另一种数据类型
- 删除：从多数据框对象或插件参数集中删除一个数据框
- 加载：加载外部文件或已保存会话
- 合并：按序列或标签合并两个数据框
- 新建：复制插件参数集，或在插件参数集中附加一个数据框，作为新的数据框对象
- 重命名：重命名数据框中的序列名称或标签名称
- 运行插件：通过运行插件执行分析
- 保存：保存会话
- 选择：根据现有数据框的筛选条件生成新的数据框

以下是一个会话示例：

1. 设置环境（将当前工作目录设置为相对于启动目录的“data”目录）

TAPPS：`1>` set rcwd data

2. 加载 CSV 文件（加载 <TAPPS 目录>/data/ STI _2015.csv，并将数据框附加为“STI”）

TAPPS：`2>` 将 csv STI 2015.csv 加载为 STI_

3. 切分数据（将“Open”从字符串类型转换为浮点数；从 STI 中切分出 STI_Low 和 STI_High 两个数据框；并将 STI_High 和 STI_Low 合并为 STI_A）

TAPPS：`3>` 将 STI 中的 Open 类型转换为非字母 TAPPS：`4>` 从 STI 中选择为 STI Low，其中 Open `< 820` TAPPS：`5>` 从 STI 中选择为 STI_High，其中 Open `> 2000` TAPPS：`6>` 从 STILow 中选择为 STI A_ TAPPS：`7>` 从 STI_High 合并标签到 STI A_

4. 设置插件参数字典（从“summarize”插件创建插件参数字典，将参数字典命名为“testingA”，并设置为对“STI_A”数据框运行“by_series”方法）

TAPPS: 8> 新建 summarize 参数 testingA TAPPS: 9> 将 testingA 中的参数分析¯名称设为 trialA TAPPS: 10> 将 testingA 中的参数分析方法¯设为 by¯series TAPPS: 11> 将 testingA 中的数据框设为 STI_A

5. 运行插件

TAPPS: 12> 运行插件 testingA

6. 附加结果数据框

TAPPS: 13> 从¯testingA 结果新建 STI 汇总数据框 TAPPS: 14> 显示数据框

当前数据框 (n = 5):

数据框名称：STI¯最高价 序列名称：Open、最高价、最低价、收盘价、成交量、调整后收盘价 序列数：6 标签数（数据行）：3919

数据框名称：STI Low ¯序列名称：Open、最高价、最低价、收盘价、成交量、调整后收盘价 序列数：6 标签数（数据行）：3

数据框名称：STI A ¯序列名称：Open、最高价、最低价、收盘价、成交量、调整后收盘价 序列数：6 标签数（数据行）：3922

数据框名称：STI 序列名称：Open、最高价、最低价、收盘价、成交量、调整后收盘价 序列数：6 标签数（数据行）：6996

数据框名称：STI summarize ¯序列名称：算术¯平均值、计数、最大值、中位数、最小值、标准_差、求和

序列数：7
标签数（数据行）：6

TAPPS: 15> 描述 STI summarize_

描述数据框 - STI_summarize

序列名称：算术平均数¯计数，最大值，中位数，最小值，标准¯差，求和 序列数：7 标签数（数据行）：6

序列名称 - 算术平均数 ¯ 算术平均数中的最小值：2673 26085131 ¯。算术平均数中的最大值：92459705 7114 ¯。字符串数据类型值的数量：0 整数数据类型值的数量：0 浮点数据类型值的数量：6 未知数据类型值的数量：0

------- <已截断> ------

7. 在退出前更改当前工作目录、保存数据框和会话（将当前工作目录更改为序列化会话的 <TAPPS 目录>/examples，将 STI_A 数据框保存为 CSV 文件，并将当前会话序列化至 <TAPPS 目录>/examples / tapps_manuscript.txt）

TAPPS: 16> 设置 ocwd TAPPS: 17> 设置 rcwd examples TAPPS: 18> 将数据框 STI A 保存为 csv¯ STI A.csv¯19> TAPPS: 将会话保存为 tapps20> manuscript.txt TAPPS: 退出

4. 结论与未来工作

本文介绍了 TAPPS，将其作为一个由领域特定命令行语言驱动的、可通过插件扩展的技术分析与应用统计工具。本文记录了 TAPPS 的架构与实现，旨在阐述进一步开发所需的基本概念。此外，本文还通过一个 TAPPS 使用会话示例进行了说明。

近期的未来工作是为 TAPPS 开发插件。例如，COPADS 中实现的假设检验程序 [16, 17] 以及现有的 Python 计量经济学方法实现 [18] 均可封装为插件。

致谢作者谨向

T. Yeo（瑞士联合银行）、J. Oon（昆士兰大学）和 C. Kuo（Yahoo）对本手稿初稿提出的意见表示感谢。

参考文献[1] R. Levesque

，《SPSS 编程与数据管理：SPSS 和 SAS 用户指南（第4版）》，SPSS 公司，2007年。[2] R. K. Meyer 和 D. D. Krueger，《Minitab 统计学指南（第3版）》，Prentice-Hall 出版社，2004年。[3] I. D. Dinov，《在线统计计算资源》，《统计软件杂志》16，2006年，第1–16页。

[4] E. Achtert、H. Kriegel 和 A. Zimek，《ELKI：用于评估子空间聚类算法的软件系统》，第20届科学与统计数据库管理国际会议（SSDBM 08）论文集，2008年。[5] J. Knight，《SOFA——人人皆可使用的开放统计软件》。Linux Journal 201，2011年，第40–41页。[6] A. R. Gilmour、R. Thompson 和 B. R. Cullis，《平均信息限制最大似然：用于线性混合模型方差参数估计的高效算法》，Biometrics 51，1995年，第1440–1450页。[7] B. Jones 和 J. Sall，《JMP 统计发现软件》，Wiley跨学科评论：计算统计学 3，2011年，第188–194页。[8] E. O. Travis，《用于科学计算的 Python》，Computing in Science & Engineering 9，2007年，第10–20页。[9] F. Morandat 和 B. Hill，《R语言设计评估：用于数据分析的对象与函数》，第26届欧洲面向对象编程会议论文集，2012年。[10] B. H. Hall 和 C. Cummins，《TSP 5.0 参考手册》，TSP international，2005年。[11] L. P. Coelho，《Mahotas：用于可编写脚本的计算机视觉的开源软件》，Journal of Open Research Software 1，2013年，3Article e。[12] S. Behrens，《使用 Python Lex-Yacc 构建解释器原型》，Dr Dobb's Journal 2004，2004年，第 30-35 页。[13] P. Mantovan 和 P. Secchi，《复杂数据建模与计算密集型统计方法》，Springer-Verlag，2010年。[14] A. Golightly 和 D. J. Wilkinson，《基于信息性观测的马尔可夫跳跃过程贝叶斯推断》，Statistical Applications in Genetics and Molecular Biology 14，2015年，第 169-188 页。[15] K. A. Seitz，《异构系统优化用字节码的设计与实现》，计算机科学荣誉论文，2012年，论文 28。[16] M. H. T. Ling，《Gopal Kanji 的 100种统计检验中的十种 Z检验例程》，The Python Papers Source Codes 1，2009年，文章 5。[17] Z. E. Chay 和 M. H. T Ling，《COPADS, II：Gopal Kanji 的 100种统计检验中的卡方检验、F检验和 t检验例程》，The Python Papers Source Codes 2，2010年，文章 3。[18] R. Bilina 和 S. Lawford，《用于计量经济学与统计学统一研究的 Python》，Econometric Reviews 31，2012年，第 558-591 页。

附录 A：适用于全部 5 种主要操作系统的统计软件包从列出的 57 个统计软件包中选出的 15 个统计软件包列表 (https://en.wikipedia.org/wiki/Comparison_of_stati

统计软件包；已访问），可用于全部 5 种主要操作系统，即 Microsoft Windows、Mac OSX、Linux、BSD 和 UNIX。在这 15 个可用于全部 5 种主要操作系统的软件包中，12 个是开源软件包。12 个开源软件包中有 4 个（DataMelt、Dataplot、SciPy 和 Statsmodels）没有明确定义的插件系统，但由于公开了内部 API，因此形成了隐式插件系统。在这 12 个开源软件包中，只有 2 个软件包（R 和 ROOT）定义了用于向系统中添加统计分析软件包的插件系统。

统计 软件包	**Open** 来源	许可证	接口	脚本编程	插件 系统
R	Yes	GNU通用公共许可证	命令行界面/图形用户界面	R、Python、Perl	Yes
ROOT	Yes	GNU通用公共许可证	GUI	C++、Python	Yes
DataMelt	Yes	GNU通用公共许可证	命令行界面/图形用户界面	Java、Jython、Groovy、Jruby、BeanShell	部分支持
Dataplot	Yes	公有领域	命令行界面/图形用户界面	None	部分支持
SciPy	Yes	BSD	CLI	Python	部分
Statsmodels	Yes	BSD	CLI	Python	部分
ADaMSoft	Yes	GNU通用公共许可证	命令行界面/图形用户界面	None	No
OpenEpi	Yes	GNU通用公共许可证	GUI	None	No
PSPP	Yes	GNU通用公共许可证	命令行界面/图形用户界面	Perl	No
Salstat	Yes	GNU通用公共许可证	命令行界面/图形用户界面	Python	No
SOCR	Yes	GNU宽通用公共许可证	GUI	None	No
SOFA Statistics	Yes	AGPL	GUI	Python	No
Statwing	No	专有	GUI	None	Yes
Brightstat	No	专有	GUI	None	No
TSP	No	专有	CLI	None	No

附录B：如何编写插件TAPPS 插件是一个 Python 模块，需遵循以下 规则：

1. 插件必须是一个文件夹，并按以下格式命名：<插件名称> <发行版本号>，其中 <插件名称>_必须是一个全小写且不含下划线或连字符的单词，而 <发行版本号> 必须是整数。例如，“generallinearmodel_1”是允许的，而“General_Linear_Model_1”、“glm_1.0”和“GeneralLinearModel_1”则不允许。2. 插件文件夹中必须包含以下文件——(1) init .py、(2) manifest.py 和 (3) main.py.3 manifest.py 中的项目必须按要求填写。4. 主文件，main.py，必须包含 3 个项目 – (1) 指令 字符串（一个多行字符串，用于向用户说明如何填写参数字典）；（2）一个名为 parameters 的字典；以及（3）一个名为 main 的函数。

5. 该 参数 字典至少必须包含以下键值对 – (1) 插件名称_ ，其必须与插件名称相对应；（2）数据框，用于将数据框作为输入附加到插件；以及（3）结果，用于让插件附加作为分析输出的数据框。以下键值对不是强制性的，但建议提供：（1）analysis_name，供用户输入对分析的简短描述；（2）分析方法，用于在存在多个分析选项时，让插件知晓要执行哪个分析函数_；以及（3）叙述，供用户输入对分析的详细描述。

6. 主函数（1）必须仅将参数字典作为唯一参数，（2）执行时只能使用参数字典中提供的信息，以保持封装，（3）必须将分析输出作为数据框返回到参数字典的结果键中，以及（4）最后必须返回参数字典。7. 主函数可以调用其他函数或其他模块。

现已提供一个插件模板（https://github.com/mauriceling/tapps/tree/master/plugins/template_1），其中包含插件所需的样板代码。鼓励插件开发者复制此文件夹，将其作为开发的基础模板。

以下代码是来自“summarize”插件（https://github.com/mauriceling/tapps/tree/master/ plugins/summarize_1）的主文件（main.py）中的一部分：

```
指令 = '''
```

```
如何填写参数字典（p）：
标准指令:
    p['分析_名称'] = <用户提供的分析名称（字符串）>
    p['叙述'] = <用户提供的分析描述（如有）>
    p['数据框'] = <来自会话的输入数据框对象['MDF'].frames>
此插件专用的指令:
    1. p['分析_方法'] 接受以下任一选项：'按_序列'（按序列 summarize）或
    '按_标签'（按标签 summarize）。'''

参数 = {'插件_名称': 'summarize', '分析_名称': None, '




def main(参数):
    '''“summarize”插件的入口函数。

    @param parameters: 参数集合，包括数据框，这是
    n插件运行所需的
    @t参数类型 ters: dicti字典
    @return: 参数
    @rtype: 字典'''
    # 步骤 1：从参数字典中提取所需的项目/数据
    方法 = 参数['分析_方法']
    数据框 = 参数['数据框']
    结果 = 参数['结果']
    # 结束第 1 步：从参数字典中提取所需项目/数据

    # 第 2 步：执行插件操作 如果 方法 == '按_序列' 或 方法 == 无:



    如果 方法 == '按_标签': (数据, 序列) =




    # 结束第 2 步：执行插件操作

    # 第 3 步：加载 da taframe 和结果写回参数字典 参数字典
    参数['数据框'] = 数据框
    参数['结果'] = 结果
    # 结束 第 3 步：将数据框和结果载入参数字典 返回 参数


定义 summarize_序列(数据框):
    ' ' '用于统计数据框中每个序列的统计摘要的函数。' '
    标签 = 数据框。数据。键()
    序列 = 数据框。序列名称_
    统计学 = [], _ [], {'求和': '算术平均数': '标准


    对于范围(len(序列))中的索引：sdata


                                                                                n
    统计学['算术_平均值'].append(算术_平均值) 索引 = int(len(sdata) / 2) # 3. 中位数 统计学['中
```

统计学['最大值'].append(sdata[-1]) 统计学['最小值'].append(sdata[0]) # 5. 最小值 统计学['计数'].

返回 (统计学, 序列)

定义 summarize_标签(数据框): '''计算数据框中每个标签的统计摘要的函数。''' 序列 = ['算术

附录C：TAPPS 语言（第 1 版）定义以下是 TAPPS 语言（第 1 版）的巴科斯-诺尔范式定义 wher保留字以粗体显示：

语句：类型转换_语句 | 删除_语句 | 描述_语句 | 加载_语句 | 合并_语句 | 新建_语句 | 重命名_语句 | 运行插件_语句 | 保存_语句 | 选择_语句 | 设置_语句 | Shell_语句 | 显示语句_

类型转换_语句：将 ID_列表中的标识符转换为指定数据类型 删除_语句：删除数据框 标识符 | 删除参数 标识符 描述语句：描述 标识符_ 加载_语句：将 CSV 文件加载为标识符 | 将无表头 CSV 文件加载为标识符 | 从文件名加载会话 合并_语句：合并序列 id_列表，从标识符到标识符 | 合并标签，从标识符到标识符 | 合并并替换标签，从标识符到标识符 新建_语句：新建标识符参数，并命名为标识符 | 从标识符新建标识符数据框 plocation 重命名语句：重命名标识符中的序列，将 id 值从一个值改为另一个值 _ | _ _ 重命名标识符中的标签，将 id_值从一个值改为 id_值 运行插件_语句：运行插件 标识符 保存_语句：将会话保存为文件名 | 将数据框 标识符 保存为 CSV 文件 文件名 选择 _ 语句：从标识符中选择，并命名为标识符 | 从标识符中选择，并命名为标识符，条件为二元运算符 值 | 从标识符中选择，并命名为标识符，条件为标识符 二元运算符 值 设置_语句：设置显示 AST 标识符 | 设置当前工作目录 文件夹 | 设置分隔符 分隔符 | 设置填充值填充选项_选项 | 在标识符中设置参数 标识符，并命名为标识符 | 在标识符中设置数据框参数，并命名为标识符 | 设置根目录 标识符 | 设置输出目录 shell 语句：Python Shell_

显示_语句：**显示 AST 历史记录** | **显示环境** | **显示历史记录** | **显示插件列表** | **显示插件** 标识符 | **显示会话** | **显示数据框** | **显示参数**
标识符_列表：全部 | 标识符 | 标识符¯列表 分隔符 标识符 值：数字_值 | 标识符¯值 二元运算符：分隔符 | 大于等于 | 小于等于 | 等于 | 不等于 数据类型：字母 | 非字母 | 浮点数 | 实数 | 整数 填入_选项：数字 | 标识符 标识符_值：标识符 | 字符串 数字 值：数字¯位置：结果 | 数据框 分隔符：分隔符 | 逗号 | 冒号 | 分号 | 反斜杠 | 竖线 | 点 | 加号 | 减号 | 乘号 | 除号 | 大于号 | 小于号

附录 D：TAPPS 字节码（第 1 版）描述下表描述了 TAPPS 字节码（第 1 版）：

操作码	操作数	描述
cast	<数据类型>， <序列名称>， <数据数据框名称>	对数据中指定的一组序列名称对应的所有数据元素进行类型转换框。
删除数据框	<数据框名称>	从多数据框对象中删除数据框。
删除参数	<参数名称>	删除插件参数字典。
描述	<数据框名称>	描述数据框——比“显示”提供更多信息。
复制数据框	<原始数据框名称>， <新数据框名称>	复制/深度复制现有数据框，并将副本附加到多数据框对象，并为其指定新的数据框名称。
贪心搜索	<数据框名称>， <新数据框名称>， <二元运算符>， <值>	从一个数据框中筛选数据并放入另一个数据框，其中一个数据元素在任何序列中，且由相同数据标签标识，满足筛选条件。
idsearch	<数据框名称>， <新数据框名称>， <序列名称>， <二元运算符运算符>， <值>	从一个数据框中选择数据到另一个数据框，其中某个数据元素在特定序列中满足筛选条件。
loadcsv1	<文件名>,<数据框名称>	加载 CSV 文件，将表头用作序列名称，并附加数据框。
loadcsv2	<文件名>,<数据框名称>	加载 CSV 文件（不含表头），并附加数据框。
loadsession	<文件名>	从文件加载之前保存的会话。
mergelabels1	<源数据框名称>,<目标数据框名称>	从以下位置合并数据，但不替换目标数据框中的任何数据：将源数据框合并到目标数据框。
mergelabels2	<源数据框名称>,<目标数据框名称>	将源数据框中的数据合并到目标数据框中。现有数据目标数据框中的数据将被源数据框中的数据替换数据框，如果在目标数据框中找到标签，则在 the源数据框中找到目标数据框中的标签。
合并序列	<序列名称>,<源数据框名称>,<目标数据框名称>	将源数据框中的一个或多个序列的数据合并到目标数据框。
新建数据框	<新数据框名称>， <参数名称>，{dataframe | 结果}	复制/深度复制输入数据框，或从某个数据框输出结果参数字典，并作为新的数据框附加。
新参数	<插件名称>， <新参数名称>	复制/深度复制已加载插件的参数字典以供使用。
Python shell	无操作数	启动 Python 交互式 Shell。
重命名序列	<数据框名称>,<原始序列名称>,<新的序列名称>	重命名数据框中的序列。
重命名标签	<数据框名称>， <原始标签名称>， <新标签名称>	重命名数据框中的标签。
运行插件	<参数名称>	使用参数字典运行插件；插件名称已编码在参数字典中。
保存为 CSV	<数据框名称>， <文件名>	将数据框保存到 CSV 文件中。
保存会话	<文件名>	将当前会话保存到文件中。
set	<变量>,<值>	在环境中或插件参数字典中设置选项。
show	<项目>	显示会话变量或环境变量/项目的值。