R 数据的导入和导出

Next: Notes

R 数据的导入和导出

这是从 R 中导入或导出数据的一个指导手册。

本文档的当前版本为 0.01 β。该文档译自 R-2.6.1 文档（2007 年 11 月 26 日）。

丁国徽（ghding@gmail.com） 译。

本文档的一些发布信息放置在 http://www.biosino.org/R/R-doc/。

ISBN 3-900051-10-0

Next: Introduction, Previous: Top, Up: Top

说明

Next: Words from the Translator, Up: Notes

版权声明

英文文档版权声明：

notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the R Development Core Team.

参考译文如下（具体以英文原文为准）：

> 版权 © 2000–2007 R Development Core Team
>
> 在遵守并包含本文档版权声明的前提下，制作和发布本文档的完整拷贝是允许的。并且，所有这些拷贝均受到本许可声明的保护。
>
> 在遵守上述完整拷贝版本有关版权声明的前提下，拷贝和发布基于本文档完整 拷贝的修改版本是允许的，并且，发布所有通过修改本文档而得到的工作成果， 须使用与本文档的许可声明一致的许可声明。
>
> 在遵守上述修改版本版权声明的前提下，拷贝和发布本文档其它语言的翻译版本 是允许的，如果本许可声明有经 R 核心开发小组（R Development Core Team）核 准的当地化译本，则遵循当地化译本。

关于本中文翻译文档的版权声明：

> 本文档为自由文档（GNU FDL），在 GNU 自由文档许可证（http://www.gnu.org/copyleft/fdl.html）下发布， 不明示或者暗示有任何保证。本文档可以自由复制，修改，散布，但请保留使用 许可声明。

Next: Acknowledgements, Previous: Copyright, Up: Notes

译者前言

不能期望一个软件可以做所有的事情[1]。R 也不例外。因此， R 需要和其它东西协作。包括我们人类，需要我们输入数据，导出数据。包括我们的 其它软件，Excel，SPSS，等等，数据格式都是特异的，需要 R 特别处理。 包括我们的数据库系统，R 不是用来管理数据的，所以需要专业的数据库帮忙。 也包括不同机器间，上面编译的 R 也是需要交换数据。这一册文档就是描述这些事情的。

R 的主要目的就是分析数据。虽然，你可以用它来处理文档，画个奥运会的鸟巢，R 的主要目的还是数据分析。这是专业化个性化的时代，特色取胜。

这个文档在几年前就写了一些。很实用的一个文档。开发人员可以大致了解 R 和其它软件大致的通讯情况，非开发人员至少可以了解， R 通过包能直接读取 Excel 表格。

已经是凌晨了，不多写了。

任何问题和建议可以给 Email！

感谢身边的朋友！

丁国徽

Email：*ghding@gmail.com*

2008 年 *1* 月 *6* 日

Previous: Words from the Translator, Up: Notes

致谢

手册中关系数据库的内容部分基于 Douglas Bates 和 Saikat DebRoy 早期写的 一个手册。本手册主要由 Brian Ripley 编写。

这里用到很多扩展包都是由自愿者贡献。这里提到的包以及主要作者如下，

CORBA	Duncan Temple Lang
foreign	Thomas Lumley, Saikat DebRoy, Douglas Bates, Duncan Murdoch and Roger Bivand
hdf5	Marcus Daniels
ncdf	David Pierce
ncvar	Juerg Schmidli
rJava	Simon Urbanek
RMySQL	David James and Saikat DebRoy
RNetCDF	Pavel Michna
RODBC	Michael Lapsley and Brian Ripley
RSPerl	Duncan Temple Lang
RSPython	Duncan Temple Lang
SJava	John Chambers and Duncan Temple Lang
XML	Duncan Temple Lang

Brian Ripley 是实现 R 的连接（connection）支持的作者。

Next: Spreadsheet-like data, Previous: Notes, Up: Top

1 绪论

尽管大多数读者觉得统计分析非常有趣，但为统计分析读入数据以及 把结果导出到其它系统以方便报表编写可能是一件比统计分析更花时间 和难办的差事。

本手册描述了 R 自身以及从 CRAN 获得的一些包里面的 数据导入和导出功能。这里描述的一些包可能还正在开发，但它们已经 提供了一些非常有用的功能了。

除非特别说明，本手册中描述的所有功能可以在各种平台运行的 R 中使用。

通常，如 R 一类的统计系统特别不适合处理大尺度的数据。 其它一些系统在这方面可以比 R 作的好。本手册的部分要点是 建议用户可以用其它系统做数据处理工作而不是用 R 里面重复的功能（ 例如，Therneau 和 Grambsch (2000) 就提到他们喜欢在 SAS 里面进行数据 处理，然后才用 S 的包 **survival** 进行数据分析）。 现在，还有几个包允许用其它编程语言（如 `Java`, `perl`

和 python）开发的函数直接整合进 R 代码里面。这样 就可以更加方便地用这些语言的功能。（见 Omegahat 项目（http://www.omegahat.org） 的 **SJava**, **RSPerl** 和 **RSPython** 包，和来自 CRAN 的 **rJava** 包）

值得注意到是 R 和 S 一样都来自 Unix 的小的可重用工具的传统， 因此，在数据导入前和结果导出后用 awk 和 perl 等 工具处理数据都是值得推崇的。 Becker, Chambers & Wilks （1988, 第 9 章）中的案例分析就是这样的 一个例子。其中，在 S 数据输入前用 Unix 工具检验和处理数据。 R 自己也是采用这种策略，比如用 perl 而不是 R 处理自身的帮助文件数据库，函数 read.fwf 开始就是调用一个 perl 代码直到后来明确在运行时不能依赖 perl。 现在，传统的 Unix 工具被很广泛的使用，包括在 Windows 系统上。

Next: Export to text files, Previous: Introduction, Up: Introduction

1.1 Imports 导入

导入 R 的数据中最容易的格式是简单的文本文件。对于小型或中型的问题， 这种格式都可以接受的。 从文本文件导入数据的原始函数（primary function）是 scan。 电子表格类似数据（Spreadsheet-like data）一章中讨论的大多数比较便利 的函数都是基于这个原始函数。

但是，所有的统计顾问们对客户用软盘或光盘提交一些私有的二进制数据 （比如，`Excel 电子表格'或`SPSS 文件'）都比较熟悉。 通常，可以做的最简单的事情是用原始软件把数据用文本文件导出（而统计 顾问们为了这个目的会在他们电脑里面安装大多数常用的软件）。不过， 这不会总是可能的[2]。 在从其它统计软件中导入数据（Importing from other statistical systems）一章中，我们会讨论一些可以在 R 里面 直接读取这些文件的工具。 对 Excel 电子表格，读取 Excel 电子表格（Reading Excel spreadsheets） 一章对可以获得的相关方法进行了总结。

在很少的一些例子中，出于简洁和快速访问考虑，数据以二进制格式保存。 这种情况下一个例子是我们已经见过几次的图像数据。它通常以二进制流的 方式保存然后在内存里面呈现，而且可能在数据前面加个信息头。 这种数据格式在二进制文件（Binary files）和二进制连接（Binary connections）部分都有所讨论。

对于大的数据库数据，通常要借助数据库管理系统（Database management system，DBMS）来处理。我们可以通过 DBMS 从数据库里面提取没有格式的 文本文件，但是对于大多数这一类型的 DBMS，我们可以直接通过 R 的包来实现数据提取操作：见关系数据库（Relational databases）部分。 通过网络连接来导入数据在网络接口（Network interfaces）一章讨论。

Next: XML, Previous: Imports, Up: Introduction

1.2 导出到文本文件中

从 R 里面导出结果通常是一个很少争论的事情，但是实际操作中仍然 有一些问题。在知道目标应用软件前提下，通常把文本文件作为最为便利的 中间转换工具。（如果需要二进制文件，见二进制文件（Binary files） 一章）。

函数 cat 是导出数据的函数的基础。它有一个 file 参数 和 append。通过连续地调用 cat

对一个文本文件写入。 最好的方式是，特别需要多次这样做的时候，首先为写入或添加文本打开一个 file 连接，然后用 cat 连接，最后关掉 （close）它。

最常见的工作是把一个矩阵或数据框以数字的矩形网格方式写入文件中， 而且还可能保留行列的标签。 这可以通过函数 write.table 和 write 来完成。 函数 write 仅可以写出一个矩阵或向量的特定列（和对一个矩阵进行转置）。 函数 write.table 更为便利，它可把一个数据框（ 或一个可以强制转换为数据框的对象）以包含行列标签的方式写出。

在把一个数据框写入到一个文本文件中时，有许多问题需要考虑。

1. 精度问题

   大多数通过这些函数对实/复数的转换是全精度的，但是用 write 时， 精度由 options(digits)的当前设置确定。 如果需要更多的控制，在一个数据框上逐列使用 format 。

2. 首行问题

   R 倾向在首行不出现表示行名字的条目，因此 在文件里面

   ```
                        dist    climb   time
   Greenmantle     2.5     650     16.083
      ...
   ```

   其它一些系统需要给行名字一个条目（可能为空），此时， 可以通过在 write.table 中 设置参数 col.names = NA 来实现。

3. 分隔符问题 文件中常用的字段分隔符是逗号，因为在英语语系的国家， 逗号几乎不可能出现在任何字段中。这种文件被称为 CSV（逗号分隔值）文件，对应的包装函数（wrapper function） write.csv 提供了适当的默认值。 在一些本地系统中，逗号作为十进制位中的小数点 （在 write.csv 函数中设置参数 dec = ","） [3] ，此时 CSV 文件以分号作为字段分隔符： write.csv2 设置了适当的默认值。

   用分号或制表符（sep = "\t"）可能是一种 比较安全的选择。

4. 缺损值问题 默认情况下，缺损值以 NA 形式输出，但这可以通过参数 na 来改变。 注意，NaN 在 write.table 里面 以 NA 看待，但在 cat 或 write 里面是区别对待的。
5. 被引号括起的字符串 默认情况下，字符串被引号括起（包括行列的名字）。 参数 quote 控制着字符和因子变量的引号引用问题。

   需要注意字符串中的引号嵌套问题。三种 有用的形式如下

   ```
   > df <- data.frame(a = I("a \" quote"))
   > write.table(df)
   "a"
   "1" "a \" quote"
   > write.table(df, qmethod = "double")
   "a"
   "1" "a "" quote"
   > write.table(df, quote = FALSE, sep = ",")
   a
   1,a " quote
   ```

   逃逸（Escape）的第二种形式常用于电子表格中。

包 **MASS** 中的函数 write.matrix 为写矩阵 提供了一种专用的接口。它同时提供了以区块方

式写的可选项，这样 可以降低内存的使用。

用 `sink` 可能把标准 R 输出重定向到一个文件中，因此 捕获了 `print` 语句（可能是暗含的）的输出。通常， 这不是最有效的办法，`options(width)`设置可能需要增加。

包 **foreign** 里面的函数 `write.foreign` 用 `write.table` 产生文本文件，同时编写一个 可以让另外一个统计包读入该文本文件的代码文件。现在支持 导出到 SPSS 和 Stata。

Previous: Export to text files, Up: Introduction

1.3 XML 文件

当从一个文本文件中读取数据时，用户有责任知道并且按习惯创建文件，比如， 在导出问文本文件（Export to text files）一节中提到的 注释字符，是否有信息头行，分隔符，缺损值的描述方式（等等）。 标签语言既可以描述内容又可以定义内容的结构，这样可以使一个文件 的内容自我明了。此时，不需要为读取这些数据的软件专门提供这些 细节信息。

可扩展标签语言（eXtensible Markup Language）– 通常简化为 XML – 可用于提供这样的结构，不仅能描述标准数据集也可以描述更复杂的数据结构。 XML 现在变得非常流行，并且作为常规数据标签和交换的标准。 它被各种团体所采用，从地理数据（如地图），图像展示[4] 到数学等。

包 **XML** 为在 R 和 S-PLUS 读写 XML 文档提供了 通用的工具。它让我们可以很容易的使用这种近年出现的技术。 多位研究人员正在探索如何在其它事情中用 XML 描述在不同应用软件中共享的数据集；存储不同系统共享的 R 和 S-PLUS 对象；通过 SVG （可扩展矢量图， Scalable Vector Graphics，XML 的一种针对矢量图应用的变种） 描述图像；描述函数文档；生成“生动的”含有文本， 数据和代码的分析/报告。

对 **XML** 包里面工具的描述已经超出本文档内容范围： 详细信息和例子见该包的主页（http://www.omegahat.org/RSXML）。 CRAN 里面的包 **StatDataML** 是基于 **XML** 包的一个例子。

Next: Importing from other statistical systems, Previous: Introduction, Up: Top

2 电子表格类似的数据

在导出为文本文件（Export to text files）一节，我们可以看到 电子表格类似的文本文件有一系列的变化样式。在这些样式中，数据以矩形 格子状呈现，而且还可能包括行列标签。在本节，我们考虑 把这种文件导入 R。

Next: Fixed-width-format files, Previous: Spreadsheet-like data, Up: Spreadsheet-like data

2.1 `read.table` 的变化样式

函数 `read.table` 是读取矩形格子状数据最为便利的方式。因为实际可能 遇到的情况比较多，所以预设了一些函数。这些函数调用了 `read.table` 但改变了它的一些默认参数。

注意，`read.table` 不是一种有效地读大数值矩阵的方法： 见下面的 `scan` 函数。

一些需要考虑到问题是：

1. 编码问题

   如果文件中包含非-ASCII 字符字段，要确保以正确的编码方式读取。 这是在 UTF-8 的本地系统里面读取 Latin-1 文件的一个主要问题。 此时，可以如下处理

   ```
   read.table(file("file.dat", encoding="latin1"))
   ```

   注意，这在任何可以呈现 Latin-1 名字的本地系统里面运行。

2. 首行问题

   我们建议你明确地设定 `header` 参数。按照惯例，首行只有对应列的字段而 没有行标签对应的字段。因此，它会比余下的行少一个字段。 （如果需要在 R 里面看到这一行，设置 `header = TRUE`。） 如果要读取的文件里面有行标签的头字段（可能是空的），以下面的方式读取

   ```
   read.table("file.dat", header = TRUE, row.names = 1)
   ```

   列名字可以通过 `col.names` 显式地设定； 显式设定的名字会替换首行里面的列名字（如果存在的话）。

3. 分隔符问题

   通常，打开文件看一下就可以确定文件所使用的字段分隔符， 但对于空白分割的文件，可以选择默认的 `sep = ""` （ 它能使用任何空白符作为分隔符，比如空格，制表符，换行符）， `sep = " "` 或者 `sep = "\t"`。 注意，分隔符的选择会影响输入的被引用的字符串。

   如果你有含有空字段的制表符分割的文件， 一定要使用 `sep = "\t"`。

4. 引用 默认情况下，字符串可以被 " 或 ' 括起，并且两种情况下，引号内部的字符都作为字符串的一部分。有效的引用字符（可能没有）的设置由 参数 `quote` 控制。对于 `sep = "\n"`， 默认值改为 `quote = ""`。

   如果没有设定分隔字符，在被引号括起的字符串里面，引号需要用 C 格式的逃逸方式逃逸，即在引号前面直接加反斜杠 \。

   如果设定了分隔符，在被引号括起的字符串里面，按照电子表格的习惯， 把引号重复两次以达到逃逸的效果。例如

   ```
   'One string isn''t two',"one more"
   ```

   可以被下面的命令读取

   ```
   read.table("testfile", sep = ",")
   ```

这在默认分隔符的文件里面不起作用。

5. 缺损值 默认情况下，文件是假定用 NA 表示缺损值， 但是，这可以通过参数 na.strings 改变。 参数 na.strings 是一个可以包括一个或多个 缺损值得字符描述方式的向量。

   数值列的空字段也被看作是缺损值。

   在数值列，值 NaN，Inf 和 -Inf 都可以被接受的。

6. 尾部空字段省略的行

   从一个电子表格中导出的文件通常会把拖尾的空字段（包括它们的分隔符） 忽略掉。为了读取这样的文件，必须设置 参数 fill = TRUE。

7. 字符字段中的空白

   如果设定了分隔符，字符字段起始和收尾处的空白会作为字段一部分看待的。 为了去掉这些空白，可以使用参数 strip.white = TRUE。

8. 空白行

   默认情况下，read.table 忽略空白行。 这可以通过设置 blank.lines.skip = FALSE 来改变。 但这个参数只有在和 fill = TRUE 共同使用时才有效。 这时，可能是用空白行表明规则数据中的缺损样本。

9. 变量的类型

   除非你采取特别的行动，read.table 将会为数据框的每个变量 选择一个合适的类型。如果字段没有缺损以及不能直接转换，它会按 logical， integer， numeric 和 complex 的 顺序依次判断字段类型。[5] 如果所有这些类型都失败了， 变量会转变成因子。

   参数 colClasses 和 as.is 提供了很大的控制权。 as.is 会 抑制字符向量转换成因子（仅仅这个功能）。 colClasses 运行为输入中的每个列设置需要的类型。

   注意，colClasses 和 as.is 对每 列专用， 而不是每个变量。因此，它对行标签列也同样适用（如果有的话）。

10. 注释

    默认情况下，read.table 用 # 作为注释标识字符。 如果碰到该字符（除了在被引用的字符串内），该行中随后的内容将会被忽略。 只含有空白和注释的行被当作空白行。

    如果确认数据文件中没有注释内容，用 comment.char = "" 会比较安全 （也可能让速度比较快）。

11. 逃逸

    许多操作系统有在文本文件中用反斜杠作为逃逸标识字符的习惯， 但是 Windows 系统是个例外（在路径名中使用反斜杠）。 在 R 里面，用户可以自行设定 这种习惯是否用于数据文件。

    read.table 和 scan 都有一个逻辑参数 allowEscapes。 从 R 2.2.0 开始，该参数默认为否，而且反斜杠是唯一被解释为 逃逸引用符的字符（在前面描述的环境中）。如果该参数设为是， 以 C 形式的逃逸规则解释，也就是控制符如 \a, \b, \f, \n, \r, \t, \v，八进制和十六进制如 \040 和 \0x2A 一样描述。任何其它逃逸字符都看着是自己，包括反斜杠。

常用函数 read.csv 和 read.delim 为 read.table 设定参数以符合英语语系本地系统中电子表格导出的 CSV 和制表符分割的文件。 这两个函数对应的变种 read.csv2 和 read.delim2 是针对在逗号作为小数点的国家使用时设计的[6]。

如果 read.table 的可选项设置不正确， 错误信息通常以下面的形式显示

```
Error in scan(file = file, what = what, sep = sep, :
        line 1 did not have 5 elements
```

或者

```
Error in read.table("files.dat", header = TRUE) :
        more columns than column names
```

这些信息可能足以找到问题所在，但是辅助函数 count.fields 可以进一步的深入研究问题所在。

读大的数据格子（data grid）时，效率最重要。设定 comment.char = ""， 以原子向量类型（逻辑型，整型，数值型，复数型，字符型或原味型）设置每列的 colClasses ，给定需要读入的行数 nrows （适当地高估一点比不设置 这个参数好）等措施会提高效率。见下面的例子

Next: Data Interchange Format (DIF), Previous: Variations on read.table, Up: Spreadsheet-like data

2.2 固定宽度格式的文件

有时，数据文件没有字段分隔符，但在预先设定的列里面含有字段的内容。 在穿孔卡片的时代，这非常普遍。现在，有时也用来节省文件空间。

函数 read.fwf 提供了一种简单的方式来读取这样的文件。 它会制定一个向量以保存字段的宽度。该函数把文件整行地读入内存， 分割结果字符串，导出一个临时的制表符分割的文件，然后调用 read.table。这对小文件已经足够了，但对于更复杂的东西，我们 推荐采用 perl 一类的编程语言对文件进行预处理。

函数 read.fortran 是处理固定格式文件的一种类似的函数。它使用 Fortran 格式的列规范。

Next: Using scan directly, Previous: Fixed-width-format files, Up: Spreadsheet-like data

2.3 数据交换模式 （DIF）

曾经用于电子表格类似的数据的旧格式是 DIF，即数据交换格式。

函数 read.DIF 提供了读取这种文件的简单方式。它 为每列设置类型的参数和 read.table 类似。

在 Windows 里面，电子表格通常在剪贴板里面以这种格式保存电子表格数据；read.DIF("clipboard") 可以直接从剪贴板里面读取数据。 这种方法比用 read.table("clipboard") 处理含有空单元的 电子表格稳健。

Next: Re-shaping data, Previous: Data Interchange Format (DIF), Up: Spreadsheet-like data

2.4 直接使用 scan 函数

read.table 和 read.fwf 都是先用 scan 读文件， 然后处理 scan 的结果。 它们非常便利，但有时直接使用 scan 效果 会比较好。

函数 scan 有很多参数。大多数参数在函数 read.table 里面 也存在。其中最为关键的参数是 what，它是用来指定从文件中读出 的变量的模式（mode）的列表。 如果该列表已经被命名，它的名字会被用作返回列表的分量。 模式可以是数值，字符或复数，并且常常用例子来指定，比如 0， "" 或 0i。例如，

```
cat("2 3 5 7", "11 13 17 19", file="ex.dat", sep="\n")
scan(file="ex.dat", what=list(x=0, y="", z=0), flush=TRUE)
```

返回一个有三个分量的列表并且丢弃 文件中的第四列。

还有一个非常有用的函数 readLines。如果你想 把所有行读入 R 然后进一步处理，可以用这个便利的函数。

scan 的一个最普遍的应用是读入大的矩阵。 假定文件 matrix.dat 只是包括一个 200 x 2000 的矩阵[7]，那么我们可用

```
A <- matrix(scan("matrix.dat", n = 200*2000), 200, 2000, byrow = TRUE)
```

在一个测试里面，这花费了 1 秒钟（在 Linux 系统测试的， 同样电脑上在 Windows 系统下则需要 3 秒钟），而

```
A <- as.matrix(read.table("matrix.dat"))
```

花费了 10 秒钟（和更多的内存），另外，

```
A <- as.matrix(read.table("matrix.dat", header = FALSE, nrows = 200,
                          comment.char = "", colClasses = "numeric"))
```

花费了 7 秒钟。造成这种差别的原因差不多完全由于 读 2000 分开的短列的时间开支所致： 如果它们的长度是 2000， scan 花费 9 秒，而 read.table 在比较高效地使用情况（特别是设定 colClasses）下需要 18 秒； 但缺乏技巧地使用 read.table 时则需要 125 秒！

注意，时限测试依赖于读的类型和数据本事。 下面是一个读取一百万个不同整数的例子：

```
writeLines(as.character((1+1e6):2e6), "ints.dat")
xi <- scan("ints.dat", what=integer(0), n=1e6)   # 0.77s
xn <- scan("ints.dat", what=numeric(0), n=1e6)   # 0.93s
xc <- scan("ints.dat", what=character(0), n=1e6) # 0.85s
xf <- as.factor(xc)                              # 2.2s
DF <- read.table("ints.dat")                     # 4.5s
```

以及一百万个小集合代码的例子：

```
code <- c("LMH", "SJC", "CHCH", "SPC", "SOM")
writeLines(sample(code, 1e6, replace=TRUE), "code.dat")
y <- scan("code.dat", what=character(0), n=1e6)  # 0.44s
yf <- as.factor(y)                               # 0.21s
DF <- read.table("code.dat")                     # 4.9s
DF <- read.table("code.dat", nrows=1e6)          # 3.6s
```

注意，这些时限测试严重依赖操作系统 （Windows 下面的基本读取所花的时间至少是 Linux 系统下面的两倍）和 垃圾收集器的精度。

2.5 数据重塑

有时，电子表格数据以一种紧凑的格式存在。它会给出各个受试者的协变量。 而每个受试者后面跟着全部的观测值。R 的建模函数需要观测值在一列内。 考虑下面来自有重复的 MRI 脑测试样本数据

```
Status   Age    V1     V2     V3    V4
     P 23646 45190  50333  55166 56271
    CC 26174 35535  38227  37911 41184
    CC 27723 25691  25712  26144 26398
    CC 27193 30949  29693  29754 30772
    CC 24370 50542  51966  54341 54273
    CC 28359 58591  58803  59435 61292
    CC 25136 45801  45389  47197 47126
```

每个受试者有两个协变量（covariate）和最多 4 个测量。 该数据从 Excel 里面导出，文件名为 mr.csv 。

我们可以用函数 stack 来帮助操作以给出唯一 的相应。

```
zz <- read.csv("mr.csv", strip.white = TRUE)
zzz <- cbind(zz[gl(nrow(zz), 1, 4*nrow(zz)), 1:2], stack(zz[, 3:6]))
```

结果为

```
     Status   Age values ind
X1        P 23646  45190  V1
X2       CC 26174  35535  V1
X3       CC 27723  25691  V1
X4       CC 27193  30949  V1
X5       CC 24370  50542  V1
X6       CC 28359  58591  V1
X7       CC 25136  45801  V1
X11       P 23646  50333  V2
...
```

函数 unstack 是相反操作的函数， 因此在导出数据时可能非常有效。

实现这个的另外一种方法是 reshape，如

```
> reshape(zz, idvar="id",timevar="var",
  varying=list(c("V1","V2","V3","V4")),direction="long")
    Status   Age var    V1 id
1.1      P 23646   1 45190  1
2.1     CC 26174   1 35535  2
3.1     CC 27723   1 25691  3
4.1     CC 27193   1 30949  4
5.1     CC 24370   1 50542  5
6.1     CC 28359   1 58591  6
7.1     CC 25136   1 45801  7
1.2      P 23646   2 50333  1
2.2     CC 26174   2 38227  2
...
```

函数 reshape 有比 stack 更为复杂的语法。但是它可以处理 比前面例子中列的数目更多的`长'格式数据。 利用 direction="wide"，reshape 也可以进行相反方向的操作。

Previous: Re-shaping data, Up: Spreadsheet-like data

2.6 无格式列联表

用数组方式展示高维列联表示很不方便的。在分类数据分析中，这种信息常常 以含有边的带行列组合的因子水平对应的单元计数的二维数组来体现。 行和列是典型的“参差”排列，因为只在它们改变时才显示标签。一个明显的习惯是， 行从顶往底部读，而列从左往右读。 在 R 里面，这种“无格式”的列联表可以用 函数 ftable 创建。 ftable 用一个适合的打印方法 创建类 "ftable" 的对象。

举个简单的例子，考虑 R 的标准数据集 UCBAdmissions。这是一个 3 维列联表， 用于对 1973 年 UC Berkeley 研究生部六个最大的系 的学生申请按照入学和性别的分类。

```
> data(UCBAdmissions)
> ftable(UCBAdmissions)
                 Dept   A   B   C   D   E   F
Admit    Gender
Admitted Male        512 353 120 138  53  22
         Female       89  17 202 131  94  24
Rejected Male        313 207 205 279 138 351
         Female       19   8 391 244 299 317
```

这种显示方式无疑比数据的 3 维数组 描述方式更有用。

还有一个函数 read.ftable 用于从文件中读取无格式的列联表。

为了处理试图准确地描述行和列变量名字和水平信息的列联表变体， 这个函数还有一些其它参数。read.ftable 的帮助页面有一些 非常有用的例子。无格式列联表可以用 as.table 转换成数组格式的标准列联表。

注意，无格式列联表的特征就是行（可能还有列）标签的“参差”排列。 如果给定行变量水平的所有格子（grid）， 应该使用函数 xtabs 从这种数据创建列联表， 而不是用 read.table 读取数据。

Next: Relational databases, Previous: Spreadsheet-like data, Up: Top

3 导入其它统计软件的数据

在本章，我们研究如何读取其它统计系统生成二进制数据文件问题。 最好避免这种问题，但若没法得到原始系统的时候，这种问题 又是不可避免的。

Next: Octave, Previous: Importing from other statistical systems, Up: Importing from other statistical

systems

3.1 EpiInfo, Minitab, S-PLUS, SAS, SPSS, Stata, Systat

推荐包 **foreign** 提供了导入这些统计系统产生的文件， 导出 Stata 或 SPSS 格式数据的工具。在一些情况下， 这些函数可能 `read.table` 需要的内存少很多。 `write.foreign` （见导出到文本文件（Export to text files）） 现在支持 SPSS 和 Stata 类型的数据导出机制。

EpiInfo 版本 5 和 6 保存的数据是自我描述的固定宽度 的文本文件。 `read.epiinfo` 可以读入这些 .REC 文件 到一个 R 数据框。 EpiData 也产生这种格式的数据。

函数 `read.mtp` 可以导入`Minitab 便携式工作表' （Minitab Portable Worksheet）文件。该函数返回一个以工作表作为分量的 R 列表。

函数 `read.xport` 读入 SAS 传输格式（XPORT）的文件，并且返回 一个数据框的列表。如果你的系统安装了 SAS， 函数 `read.ssd` 可用来创建和运行以传输格式保存 SAS 永久数据集（.ssd 或 .sas7bdat）的 SAS 脚本。 它随后调用 `read.xport` 去读取 结果文件。包 **Hmisc** 有个类似的函数 `sas.get`，它 也是允许 SAS 脚本。

函数 `read.S` 可以读取（32 位）Unix 或 Windows（或其它操作系统） 上由 S-PLUS 3.x，4.x 或 2000 产生二进制对象。这能读取许多但不是全部的 S 对象：特别是，它只能读取向量，矩阵，数据框和含有这类数据对象的 列表。

函数 `data.restore` 用于读 S-PLUS 的转储数据（data dump） （由 `data.dump` 创建）。它有同样的限制（除了 α 平台 的转储数据也可被读取）。它还可能读取来自 S-PLUS 5.x 和 6.x 通过 `data.dump(oldStyle=T)` 写出的转储数据。

如果可以访问 S-PLUS，更可靠的方式是在 S-PLUS 里面导出 （`dump`）对象文件然后在 R 里面载入执行（`source`） 该文件。在 S-PLUS 5.x 和 6.x 里面，需要用 `dump(..., oldStyle=T)`，对于读入大对象， 优先使用用转储文件作为批量的脚本而非 `source`。

函数 `read.spss` 可以读取 SPSS 里面 `save' 和 `export' 命令创建的文件。 它返回一个由被保存数据集中每个变量对应分量的列表。 含有值标签的 SPSS 变量可以选择转换为 R 因子。

SPSS 数据入口（Data Entry）是创建数据输入的窗体。 默认情况下，它创建一种 `read.spss` 不能处理的 含有额外格式信息的数据文件。但是，它可能以普通的 SPSS 格式导出数据。

Stata 的 .dta 文件是二进制文件格式。 函数 `read.dta` 和 `write.dta` 可以读写版本 5，6，7/SE 和 8 的 Stata 文件。 有值标签的 Stata 变量可以选择性地转换为 R 因子（反之也行）。

函数 `read.systat` 可以读取 Systat 在小字节序机器（little-endian machines）（比如 Windows）上保存（`SAVE`）的 矩形的数据文件（`mtype = 1`）。 这些文件的扩展名为 .sys 或 .syd （最近）。

Previous: EpiInfo Minitab SAS S-PLUS SPSS Stata Systat, Up: Importing from other statistical systems

3.2 Octave

Octave 是一种线性代数的数值运算系统（http://www.octave.org）， 来自包 **foreign** 的 函数 `read.octave` 可以读入 Octave 用命令 save -ascii 创建的文本数据格式。 该格式支持变量的大多数通用类型，包括标准的原子型（实和复标量/矩阵， 和 N 维数组，字符串，极差

（range），和布尔值标量/矩阵）和 递归式（结构体（structs），单元（cells）和列表）。

Next: Binary files, Previous: Importing from other statistical systems, Up: Top

4 关系数据库

Next: Overview of RDBMSs, Previous: Relational databases, Up: Relational databases

4.1 为什么使用数据库？

R 可以很好处理的数据类型是有限的。既然所有 R 处理的数据都放在 内存中，并且在一个函数的执行过程中会创建一个数据集的多个拷贝， 因此 R 不适合处理很大的数据集。 大于一百兆（或少一点）的数据对象会导致 R 的内存溢出。

R 不容易支持并发访问数据。也就是说，如果多于一个用户在访问 或者更新同一个数据，一个用户的修改对另外一个用户是不可见的。

R 支持数据的永久性，因为用户需要从一个回话中保存 数据对象或整个工作表而在随后的会话中再次保存。 但被保存数据的格式对 R 是特有的，在其它系统里面不是 那么容易被处理。

数据库管理系统（Database Management Systems，DBMSs）和，特别是， 关系数据库管理系统（RDBMSs）是为了更好地做这些事情而设计。. 它们加强的地方在于

1. 提供对大型数据库中被选择部分的快速访问。
2. 强大的针对数据库的列进行汇总和交叉列表（cross-tabulate）的功能。
3. 以更有条理的方式存储数据。这比电子表和 R 的数据框 的矩形网格格式更容易组织。
4. 支持运行于不同主机上面的客户端的并发访问，同时 加强数据访问的安全性限制。
5. 有能力作为一个服务很多客户端的服务器。

DBMS 可能要用到的统计应用的类型是从数据中 提取 1%的样本，对数据交差列表（cross-tabulate）以产生一个 多维的列联表，和为单独的分析从数据库中提取数据组。

Next: R interface packages, Previous: Why use a database?, Up: Relational databases

4.2 对 RDBMs 的回顾

已经有很多大型的（也是很贵的）商业的关系数据库管理系统（ Informix； Oracle; Sybase； IBM 的 DB/2；Microsoft 运行在 Windows 系统上的 SQL Server）和学术的小型系统的数据（如 MySQL，PostgreSQL， Microsoft Access，...）。前者都会强调数据安全性特征。 但是界限是很模糊的，比如开源的 PostgreSQL 有越来越多的高端特性 [8]， 以及`免费'的 Informix，Oracle 和 Sysbase 在 Linux 系统下面都可以 获得。

还有其它常常使用的数据源，包括电子表格， 非关系型数据库，甚至文本文件（可能已经压缩过的）。 开放数据库互连（Open Database Connectivity，简写为 ODBC） 是使用所有这些数据

源的标准。它源于 Windows 系统 （见 http://www.microsoft.com/data/odbc/），但 Linux/Unix 系统 也实现这个标准。

本章后面描述的所有包都提供了客户端/服务器数据库的客户端。 数据库可以放置在一样的机器上或远程（这更常见）。 数据库交互时有一个 ISO 标准（事实上有好多种： SQL92 就是 ISO/IEC 9075，也被称为 ANSI X3.135-1992，此外 SQL99 也逐步被使用了）的 的交互语言 SQL （结构化查询语言，Structured Query Language， 有时读作`sequel'：见 Bowman *et al.* 1996 以及 Kline 和 Kline 2001）。不同的 DBMSs 对这个标准都是在一定程度上支持。

- SQL queries: SQL 查询
- Data types: 数据类型

Next: Data types, Previous: Overview of RDBMSs, Up: Overview of RDBMSs

4.2.1 SQL 查询

对于常规操作，非常全面的 R 接口可以产生 SQL，但是对于复杂 操作，只有直接使用 SQL。习惯上，SQL 用大写字母编写， 但是很多用户发现在 R 接口函数里面用 小写比较方便。

一个关系型 DBMS 以 表格（tables） （或 关系（relations））数据库的方式存储数据。数据库的表格 和 R 的数据框类似，因为它们都是由同一类型（数值，字符，日期，货币，...）的 列（column）或者 字段（fields） 和 包含实体观测数据的 行（row）或 记录（record）组成。

SQL `查询'是关系数据库里面最常用的操作。 典型的查询是下面类型的 SELECT 语句

```
SELECT State, Murder FROM USArrests WHERE Rape > 30 ORDER BY Murder

SELECT t.sch, c.meanses, t.sex, t.achieve
  FROM student as t, school as c WHERE t.sch = c.id

SELECT sex, COUNT(*) FROM student GROUP BY sex

SELECT sch, AVG(sestat) FROM student GROUP BY sch LIMIT 10
```

上面语句的第一条从已经复制到一个数据库表的 R 数据框 `USArrests` 数据中选择两列， 基于第三列做子集操作，并且让结果数据排序。 第二条语句连接（join）两个表格 `student` 和 `school`，随后返回四列。第三和第四个查询 进行了一些交差列表操作然后返回计数和平均值。 （五个聚合函数分别是 COUNT(*) 以及用于列的 SUM， MAX， MIN 和 AVG）

SELECT 查询用 FROM 选择表，WHERE 设定查询条件（或者被 AND 或 OR 分割的多个条件），然后用 ORDER BY 对结果进行排序。 和数据框不一样的是，RDBMS 表里面的行最好看作是无序的， 如果没有 ORDER BY 语句，顺序是不确定的。 你可以对对不止一列进行排序（辞典排序方式）， 各个列间用逗号分隔。 把 DESC 放在 ORDER BY 后面会让结果按降序排列。

SELECT DISTINCT 查询只返回被选中的表中 不同的行。

GROUP BY 字句选择基于标准的行的子集。 如果不止一列被设定（逗号分隔），可以用五个聚合函数 中的一个对多维交差分类的数据进行汇总。 HAVING 字句可以基于聚合值选择或去掉一些组。

如果 SELECT 语句含有产生唯一排序的 ORDER BY 语句， LIMIT 字句可加入选择（通过数字）输出行的一个连续的区块。 在同时取得一个区块行时，该语句非常有用。 （除非排序是

唯一的否则它可能不可靠，因为 LIMIT 字句 可用来优化查询。）

还有查询用于创建表（CREATE TABLE，但通常是利用这些接口 把一个数据框复制到数据库），INSERT（插入）或 DELETE（删除） 或 UPDATE（更新）数据。可用 DROP TABLE ` 查询'破坏一个表 [9]。

Kline 和 Kline (2001) 对 SQL Server 2000，Oracle，MySQL 和 PostgreSQL 里面 SQL 语句的实现细节进行了论述。

Previous: SQL queries, Up: Overview of RDBMSs

4.2.2 数据类型

数据可用任何一种数据类型保存在数据库里面。数据类型的范围 是依赖于 DBMS，但是 SQL 标准定义了许多类型，包括下面广泛 实现的类型（不一定采用 SQL 的名字）。

`float(`*p*`)`
: 实数，精度可选。常常被称着 `real` 或 `double` 或 `double precision`。

`integer`
: 32 位整数。常常称为 `int`。

`smallint`
: 16 位整数

`character(`*n*`)`
: 固定长度的字符串。常常称为 `char`。

`character varying(`*n*`)`
: 可变长度的字符串。常常称为 `varchar`。 几乎总有一个上限 255 个字符串。

`boolean`
: true 或者 false。有时被称为 `bool` 或 `bit`。

`date`
: 历法日期

`time`
: 当前时间

`timestamp`
: 日期和时间

因为时区（`with timezone`）的缘故，`time` 和 `timestamp` 有很多变种。 其它广泛实现类型是用保存大块头的文本和二进制数据的 `text` 和 `blob`。

全面实现的 R 接口包为用户掩藏了很多 类型转换的问题。

Previous: Overview of RDBMSs, Up: Relational databases

4.3 R 的接口包

CRAN 已经有几个帮助 R 连接 DBMS 的包。 它们提供了不同层次上的抽象。 一些包为整个数据框和数据库间的数据复制提供方法。 所有的包都提供了通过 SQL 查询从数据库里面 选择数据，以数据框获得全部结果数据或者部分（通常是行 的组）的功能。

除 **RODBC** 外的所有包都是连到一个 DBMS。统一`前端'包 **DBI** （http://developer.r-

project.org/db）和`后端'开发最完善的 包 **RMySQL** 的工作现在正在进行。 同样在 CRAN，还有后端包 **ROracle** 和 **RSQLite** （该项目和 SQLite 数据库管理系统同时在开发，http://www.hwaci.com/sw/sqlite）。

两个早期的包 **RmSQL** 和 **RPgSQL** 现在已经不再支持， 已经属于 CRAN 的开发范围： BioConductor 项目有包 **RdbiPgSQL**。 **PL/R** （http://www.joeconway.com/plr/）是一个把 R 嵌入 PostgreSQL 的项目。

- DBI / RMySQL: DBI / RMySQL
- RODBC: RODBC

Next: RODBC, Previous: R interface packages, Up: R interface packages

4.3.1 包 DBI 和 RMySQL

CRAN 的包 **RMySQL** 提供了 MySQL 数据库系统的接口（见 http://www.mysql.com 和 Dubois，2000）。 这里的描述用于版本 `0.5-0`：早期版本有很多不一样的接口。 当前版本需要 **DBI**，这里的描述通过少量修改也可用于其它 支持 **DBI** 包的后端。

MySQL 存在于 Unix/Linux 和 Windows 上面：从 3.23.x 版本开始（2001 年 1 月）， 它以 GPL 协议发布。MySQL 是一个`轻量级'（light and lean）的数据库。 （在大小写敏感的文件操作系统，它支持名字的大小写。注意，Windows 系统里面文件名大小写 不敏感。）[10] 包 **RMySQL** 在 Linux 和 Windows 系统上都可以使用。

`dbDriver("MySQL")` 的调用会返回一个数据库连接管理对象，然后 调用 `dbConnect` 打开一个数据库连接，随后会调用泛型函数 `dbDisconnect` 来关闭这个连接。 对于，ORACLE 和 SQLITE 系统，分别使用 **ROracle** 或 **RSQLite** 里面的 `dbDriver("Oracle")` 函数或 `dbDriver("SQLite")` 函数。

SQL 查询可以通过 `dbSendQuery` 或 `dbGetQuery` 传给数据库管理系统。`dbGetQuery` 传送查询语句， 把结果以数据框形式返回。`dbSendQuery` 传送查询，返回的结果是 继承`"DBIResult"`的一个子类的对象。`"DBIResult"` 类 可用于取得结果，而且还可以通过调用 `dbClearResult` 清除结果。

函数 `fetch` 用于获得查询结果的部分或全部行，并以列表返回。 函数 `dbHasCompleted` 确定是否所有行已经获得了， 而 `dbGetRowCount` 返回结果中行的数目。

这些是数据库中读/写/测试/删除表的方便接口。 `dbReadTable` 和 `dbWriteTable` 实现一个 R 数据框的复制进和复制出数据库， 把数据框的行名字映射到 MySQL 表的 `row_names` 字段。

```
> library(RMySQL) # will load DBI as well
## 打开一个MySQL数据库的连接
> con <- dbConnect(dbDriver("MySQL"), dbname = "test")
## 列出数据库中表
> dbListTables(con)
## 把一个数据框导入到数据库，删除任何已经存在的拷贝
> data(USArrests)
> dbWriteTable(con, "arrests", USArrests, overwrite = TRUE)
TRUE
> dbListTables(con)
[1] "arrests"
## 获得整个表
> dbReadTable(con, "arrests")
```

```
               Murder Assault UrbanPop Rape
Alabama          13.2     236       58 21.2
Alaska           10.0     263       48 44.5
Arizona           8.1     294       80 31.0
Arkansas          8.8     190       50 19.5
...
## 从导入的表中查询
> dbGetQuery(con, paste("select row_names, Murder from arrests",
                        "where Rape > 30 order by Murder"))
   row_names Murder
1   Colorado    7.9
2    Arizona    8.1
3 California    9.0
4     Alaska   10.0
5 New Mexico   11.4
6   Michigan   12.1
7     Nevada   12.2
8    Florida   15.4
> dbRemoveTable(con, "arrests")
> dbDisconnect(con)
```

Previous: DBI / RMySQL, Up: R interface packages

4.3.2 包 RODBC

CRAN 里面的包 **RODBC** 提供了支持 ODBC 规范的 访问数据源的接口。它有广泛的需求，可以使一样的 R 代码访问不同的 数据库系统。 在 Unix/Linux 以及 Windows 系统都能运行包 **RODBC**，而且几乎 所有的数据库系统都支持 ODBC。我们已经在 Windows 平台上的 Microsoft SQL Server，Access，MySQL 和 PostgreSQL 以及 Linux 平台上的 MySQL，Oracle，PostgreSQL 和 SQLite 都测试过了。

ODBC 是一种客户端-服务器系统。我们可以从 Windows 客户端 连接 Unix 服务器上运行的 DBMS，反之也然。

Windows 通常都会支持 ODBC，最新版本作为 MDAC 的一部分可以从 http://www.microsoft.com/data/odbc/ 下载。 在 Unix/Linux 系统，你需要一个 ODBC 驱动管理，比如 unixODBC (http://www.unixODBC.org) 或 iOBDC (http://www.iODBC.org)，还需要为你的数据库系统 安装一个驱动。FreeODBC 项目 （http://www.jepstone.net/FreeODBC/）是和 ODBC 信息相关的一个知识库。

Windows 不仅为 DBMSs 提供驱动， 还为 Excel 电子表格（.xls），DBase（.dbf）文件甚至文本文件 提供驱动。（不需要安装命名过的软件。）现在已经有 Excel 2007 和 Access 2007 版本的驱动了（去 http://download.microsoft.com，选择`Office'并且 找到`ODBC'，然后下载 AccessDatabaseEngine.exe）。

大量同时访问是可能的。调用 `odbcConnect` 或 `odbcDriverConnect` （在 Windows 图形化界面下，可以通过对话框选择数据库） 可以打开一个连接，返回一个用于随后数据库访问的控制（handle）。 打印一个连接会给出 ODBC 连接的一些细节，而调用 `odbcGetInfo` 会给出客户端和服务器的一些细节信息。

连接可以通过调用函数 `close` 或 `odbcClose` 来关闭。 没有 R 对象对应或不在 R 会话后面的连接也可以调用这两个函数来关闭， 但会有警告信息。

在一个连接中的表的细节信息可以通过函数 sqlTables 获得。

函数 sqlSave 会把 R 数据框复制到一个数据库的表中， 而函数 sqlFetch 会把一个数据库中的表拷贝到 一个 R 的数据框中。

一个 SQL 查询可以通过调用 sqlQuery 传给 数据库。返回的结果是 R 的数据框。（sqlCopy 把一个 查询传给数据库，返回结果在数据库中以表的方式保存。） 一种比较好的控制方式是首先调用 odbcQuery， 然后 用 sqlGetResults 取得结果。后者可用于一个循环中 每次获得有限行，就如函数 sqlFetchMore 的功能。

这里是用 PostgreSQL 的一个例子，其中 ODBC 驱动 把列和数据框的名字映射成小写。我们用一个事先创建的数据库 testdb，还有一个个设置在 unixODBC 下文件 ~/.odbc.ini 的数据源名字（Data Source Name，DSN）。同样的代码在 MyODBC 访问 Linux 或 Windows 上的 MySQL 数据库时一样有效（其中，MySQL 依然会把名字 映射成小写）。在 Windows，DSN 在控制面板的 ODBC 工具里面设置（在 Windows 2000/XP，设置`管理工具'部分的`数据源（ODBC）'）。

```
> library(RODBC)
## 让函数把名字映射成小写
> channel <- odbcConnect("testdb", uid="ripley", case="tolower")
## 把一个数据框导入数据库
> data(USArrests)
> sqlSave(channel, USArrests, rownames = "state", addPK = TRUE)
> rm(USArrests)
## 列出数据库的表
> sqlTables(channel)
  TABLE_QUALIFIER TABLE_OWNER TABLE_NAME TABLE_TYPE REMARKS
1                              usarrests      TABLE
## 列出表格
> sqlFetch(channel, "USArrests", rownames = "state")
               murder assault urbanpop rape
Alabama          13.2     236       58 21.2
Alaska           10.0     263       48 44.5
    ...
## SQL 查询，原先是在一行的
> sqlQuery(channel, "select state, murder from USArrests
           where rape > 30 order by murder")
       state murder
1 Colorado      7.9
2 Arizona       8.1
3 California    9.0
4 Alaska       10.0
5 New Mexico   11.4
6 Michigan     12.1
7 Nevada       12.2
8 Florida      15.4
## 删除表
> sqlDrop(channel, "USArrests")
## 关闭连接
> odbcClose(channel)
```

作为 Windows 下面用 ODBC 连接 Excel 电子表格的一个简单例子， 我们可以如下读取电子表格

```
> library(RODBC)
> channel <- odbcConnectExcel("bdr.xls")
## 列出电子表格
```

```
> sqlTables(channel)
  TABLE_CAT TABLE_SCHEM        TABLE_NAME   TABLE_TYPE REMARKS
1 C:\\bdr            NA           Sheet1$ SYSTEM TABLE      NA
2 C:\\bdr            NA           Sheet2$ SYSTEM TABLE      NA
3 C:\\bdr            NA           Sheet3$ SYSTEM TABLE      NA
4 C:\\bdr            NA Sheet1$Print_Area        TABLE      NA
## 获得表单1的内容，可以用下面任何一种方式
> sh1 <- sqlFetch(channel, "Sheet1")
> sh1 <- sqlQuery(channel, "select * from [Sheet1$]")
```

注意，数据库表的规范和 `sqlTables` 返回的名字是不一样的： `sqlFetch` 可以映射这种差异。 library(RPgSQL) --> db.connect(dbname="testdb") # add authentication as needed --> data(USArrests) --> usarrests <- USArrests --> names(usarrests) <- tolower(names(USArrests)) --> db.write.table(USArrests, write.row.names = TRUE) --> db.write.table(usarrests, write.row.names = TRUE) --> rm(USArrests, usarrests) --> db.ls() --> db.read.table("USArrests") --> bind.db.proxy("USArrests") --> USArrests[, "Rape"] --> rm(USArrests) # remove proxy --> db.execute("SELECT rpgsql_row_names, murder FROM usarrests", --> db.fetch.result() --> db.rm("USArrests", "usarrests") # use ask=FALSE to skip confirmation --> db.ls() --> db.disconnect() -->

Next: Connections, Previous: Relational databases, Up: Top

5 二进制文件

- Binary data formats: 二进制数据格式
- dBase files (DBF): DBase 文件（DBF）

二进制连接（Connections）是现在首选的 处理二进制文件的方法。

Next: dBase files (DBF), Previous: Binary files, Up: Binary files

5.1 Binary data formats 二进制数据格式

包 **hdf5**，**RNetCDF** 和 **ncdf** 为 NASA 的 HDF5 （层次数据格式，Hierarchical Data Format， 见 http://hdf.ncsa.uiuc.edu/HDF5/）和 UCAR 的 netCDF 数据文件 （网络公共数据格式，network Common Data Form，见 http://www.unidata.ucar.edu/packages/netcdf/）提供了接口。

二者都是用面向数组的方式存储科学数据的系统，包括 描述，标签，格式，单位，...。HDF5 允许 成组的数组，因此 R 的接口会把列表 映射成 HDF5 的组，并且可以写数值和字符向量/矩阵。

CRAN 里面的包 **ncvar** 通过 包 **RNetCDF** 为 netCDF 数据文件提供高水平的 R 接口。

还有一个来自 http://www.bioconductor.org 的包 **rhdf5**。

Previous: Binary data formats, Up: Binary files

5.2 dBase 文件（DBF）

dBase 是 Ashton-Tate 写的一个 DOS 程序，随后被 Borland 拥有。它有一种很流行的二进制无格式文件，以 .dbf 作为扩展名。它被 'Xbase' 家族的数据库 （包括 dBase，Clipper，FoxPro 和它们的 Windows 的对应物 Visual dBase，Visual Objects 和 Visual FoxPro，见 http://www.e-bachmann.dk/docs/xbase.htm）广泛采用。 dBase 文件含有一个信息头，随后是一系列字段， 因此和 R 的数据框非常类似。 数据本身以文本格式保存，可以包括 字符，逻辑和数值字段，和以后版本的其它类型 （见 http://clicketyclick.dk/docs/data_types.html）。

函数 `read.dbf` 和 `write.dbf` 提供了在 所有 R 平台上读和写基本 DBF 文件的途径。 **RODBC** 包里面的函数 `odbcConnectDbase` 通过 Microsoft 的 dBase ODBC 驱动为 Windows 用户 提供了更为全面的工具读取 DBF 文件（Visual FoxPro 驱动可以 通过 `odbcDriverConnect` 使用）。

Next: Network interfaces, Previous: Binary files, Up: Top

6 连接

连接（Connections）在 R 里面的使用是基于 Chambers (1998)的建议。 它是用一组函数去实现灵活的指向类似文件对象的接口代替文件名的使用。 [11]

Next: Output to connections, Previous: Connections, Up: Connections

6.1 连接类型

最熟悉的连接类型是文件，而文件连接由函数 `file` 创建。 文件连接打开后可以在文本或二进制模式下读写添加文本（ 如果操作系统允许的话）。 事实上，文件可以为读和写同时打开，并且 R 为读和写保持分离的文件位置。

注意，一个连接创建后默认不是打开的。 基本原则是使用连接的函数在连接还没有打开时 需要打开一个连接（必须的），并且如果它打开了一个连接 在使用后需要关闭它。简单来说，让连接处于你发现它时的状态。 有泛型函数 `open` 和 `close` 的方法去显式地打开或关闭连接。

`gzip` 算法压缩的文件可以通过函数 `gzfile` 创建连接，而 `bzip2` 算法压缩的文件通过 `bzfile` 创建连接。

Unix 程序员习惯用 `stdin`， `stdout` 和 `stderr` 处理特定的文件。 这些以 终端连接（Terminal Connection）的 形式在 R 里面存在。它们可能是通常的文件，但是它们还可能 指向从图形化控制台输入或输出的内容。（尽管使用标准的 Unix R 接口，`stdin` 使用通过 `readline` 提交的行 而不是一个文件。）

这三个终端连接总是打开的，并且不能再打开或关闭。 `stdout` 和 `stderr` 习惯用于 正常输出（标准输出流）和错误信息的输出（标准错误流）。 它们可能输出到同一个地方，但是尽管

正常输出可以通过 调用函数 sink 实现重定向，传送给 stderr 的 错误信息输出必须使用 sink, type="message")。 需要特别注意这里使用的语句：连接不能重定向，但输出可以发送给其它连接。

文本连接（Text connections）是另外一种输入源。 它们允许 R 读入的字符向量就像从文本文件中读入的行。 可用调用 textConnection 来创建和打开文本连接。 函数 textConnection 在创建文本连接的时候把 字符向量的当前内容复制到内部缓存里面。

文本连接可以用来捕获 R 输出到一个字符向量中。 textConnection 也能用来 在用户的工作空间创建一个新的字符对象或者 添加到一个已有的对象后面。 通过调用 textConnection 打开连接， 并且任何时候输出到连接的全部行都可以 从 R 对象获得。关闭连接时会把所有余下的输出 写入到字符向量的最后一个元素。

管道（Pipes）是连接到其它进程的文件的一种特有形式， 管道连接通过函数 pipe 创建。 为写而打开一个管道连接（在管道后面添加内容是没有意义的）时， 首先运行一个操作系统命令，然后把标准输出和 R 连接， 最后把内容写入到那个管道。相反，为输入打开一个管道连接， 也是运行一个操作系统命令，然后让它的标准输出作为 R 从那个连接获得的输入。

URL 类型的 http://，ftp:// 和 file:// 可以通过函数 url 读内容。为方便起见，file 也可以 接受这种文件规范和调用 url。

套接字（Sockets）在支持 Berkeley 类型的套接字系统 （大部分 Unix 系统，Linux 和 Windows）上可以通过 函数 socketConnection 创建连接。套接字可以写入也可以读入， 并且客户端和服务器端的套接字都可以使用。

Next: Input from connections, Previous: Types of connections, Up: Connections

6.2 输出到连接

我们已经描述过函数 cat，write，write.table 和 sink 的写入文件的功能，特别是参数 append = TRUE 时， 在一个文件末尾添加内容的功能。这些也是这些函数在 R 1.2.0 之前的功能。

这些函数当前功能的体现和以前是一样的，但是实际发生的是， 当 file 参数是一个字符串时，打开文件连接（写入或者添加）和 函数调用后再次关闭连接。 如果我们想重复性地写入到同样的文件中， 显式地声明和打开一个连接，把这个连接对象传递给 一个输出函数是比较有效的。这使得写入到一个管道 成为一种可能。在早期的版本中，通过语法 file = "|cmd" 已经部分实现（现在还在使用）。

还有一个函数 writeLines 用于把全部的文本行 写入到一个连接。

一些简单的例子如下

```
zz <- file("ex.data", "w")  # 打开一个输出文件连接
cat("TITLE extra line", "2 3 5 7", "", "11 13 17",
    file = zz, sep = "\n")
cat("One more line\n", file = zz)
close(zz)

## 使用管道（Unix）在输出中把小数点转换成逗号
## R字符串和（可能）SHELL脚本中都需要把 \ 写两次
zz <- pipe(paste("sed s/\\\\./,/ >", "outfile"), "w")
cat(format(round(rnorm(100), 4)), sep = "\n", file = zz)
close(zz)
```

```
## 现在查看输出文件：
file.show("outfile", delete.file = TRUE)

## 捕获R输出：使用 help(lm) 里面的例子
zz <- textConnection("ex.lm.out", "w")
sink(zz)
example(lm, prompt.echo = "> ")
sink()
close(zz)
## 现在 `ex.lm.out' 含有需要进一步处理的输出内容
## 查看里面的内容，如
cat(ex.lm.out, sep = "\n")
```

Next: Listing and manipulating connections, Previous: Output to connections, Up: Connections

6.3 从连接输入

从连接读入数据的基本函数是 scan 和 readLines。这些函数有个以字符串作为输入的参数，在 函数调用时会打开一个文件连接，但显式地打开文件连接允许一个文件 可以连续地以不同格式读入。

调用 scan 的其它函数也可以使用连接， 特别是 read.table。

一些简单的例子如下

```
## 读入前面例子中创建的文件
readLines("ex.data")
unlink("ex.data")

## 读入当前目录的清单（Unix）
readLines(pipe("ls -1"))

# 从输入文件中去掉拖尾的逗号。
# 假定我们有一个包含如下`数据'的文件
450, 390, 467, 654,  30, 542, 334, 432, 421,
357, 497, 493, 550, 549, 467, 575, 578, 342,
446, 547, 534, 495, 979, 479
# 然后通过如下命令读入
scan(pipe("sed -e s/,$// data"), sep=",")
```

为方便起见，如果 file 的参数指定的是 FTP 或 HTTP URL， 则该 URL 会通过函数 url 打开读入内容。 通过 file://foo.bar 指定文件也是允许的。

- Pushback: 压栈

Previous: Input from connections, Up: Input from connections

6.3.1 压栈

C 程序员可能对 ungetc 函数非常熟悉。这个函数会把一个字符退回到 文本输入流中。R 连接以一种更为强大的方式实现一样的想法，函数 pushBack 可以把任意行的文本（本质上）压入给连接。

压栈操作类似堆栈，因此一个读请求首先使用从最近压入的文本行， 然后才是早期压入的行，最后读连接自己。 一旦一个压入行已经读完，它会被清除掉。通过函数 pushBackLength 可以查看处理中的压入行当数目。 下面是一个简单的例子

```
> zz <- textConnection(LETTERS)
> readLines(zz, 2)
[1] "A" "B"
> scan(zz, "", 4)
Read 4 items
[1] "C" "D" "E" "F"
> pushBack(c("aa", "bb"), zz)
> scan(zz, "", 4)
Read 4 items
[1] "aa" "bb" "G"  "H"
> close(zz)
```

压栈操作仅适用于文本输入模式的连接。

Next: Binary connections, Previous: Input from connections, Up: Connections

6.4 列出和操作连接

通过函数 showConnections() 可以汇总当前用户打开的连接。 通过函数 showConnections(all = TRUE) 则可以查看所有连接的汇总信息，包括已经关闭或终止的连接。

泛型函数 seek 用于读和（在一些连接里） 重新设置读或写的当前位置。不幸的是，它依赖于操作系统， 因此可能不太可靠（如，在 Windows 下面处理文本文件）。 函数 isSeekable 判断 seek 是否可以修改 由参数设定的连接的位置。

函数 truncate 可用于截去为在当前位置写而打开的文件。 它仅用于 file 连接，并且不是在所有平台都可以用。

Previous: Listing and manipulating connections, Up: Connections

6.5 二进制连接

函数 readBin 和 writeBin 可以读写二进制连接。 二进制模式打开的连接可以通过添加 "b" 的方式设置 读写规范，即用 "rb" 表示读，"wb" 或 "ab" （如果可以的话）表示写。这些函数拥有的参数如下

```
readBin(con, what, n = 1, size = NA, endian = .Platform$endian)
writeBin(object, con, size = NA, endian = .Platform$endian)
```

在两种情况下，con 是一个在函数调用过程中因需要而打开的 连接，如果给定的是字符串，它会被假定是文件名字。

描述输出略微简单一点，因此我们首先描述它。 object 必须是一个原子型向量对象，也就是没有属性的 numeric，integer，logical，character， complex 或 raw 模式的向量。默认情况下，这些以 和内存里面完全一样字节流的写入文件的。

readBin 从文件中读入字节流，把它们解释为 what 给定 模式的向量。这既可以是一个适当

模式（比如，what=integer()）的 对象，也可以是字符串所描述的模式（前面章节给出的五种模式中的一种， 或"double" 或 "int"）的对象。参数 n 指定 从连接中读入向量元素的最大数目： 如果可以获得的元素比较少，会返回一个短的向量。 参数 signed 允许单字节和双字节整数 作为有符号（默认）或无符号整数读入。

剩下的两个参数用于和其它程序或平台交换数据而读写数据。 默认情况下，二进制数据直接从内存传给连接，反之亦然。 在不同体系架构的机器间传输文件时，这两个参数是不够的，但是， 几乎所有的 R 平台之间仅仅需要改变字节序（byte-order）。 普通 PC（基于 ix86 和 x86_64 的机器），Compaq Alpha 和 Vaxen 是小字节序（little-endian）的，Sun Sparc，mc680x0 系列，IBM R6000， Apple Macintosh， SGI 和许多其它电脑都是 大字节序（big-endian） 的。（网络字节序（如 XDR[12]）是大字节序的。） 转换来自其它程序的数据，我们可能需要做更多的事情，比如，读 16 位的整数 或写单精度的实数。 这可以通过 size 参数来做。这个参数（通常）允许整数 和逻辑值的存储大小为 1，2，4，8，允许实数存储大小为 4，8 和 12 或 16（可能的话）。 在不同存储大小之间转换可能会丢失很多精度， 并且不能用于含 NA 的向量。

字符串以 C 格式读写，这是一个以零字节结束的字节流。 函数 readChar 和 writeChar 提供了更大的灵活性。

- Special values: 特殊值

Previous: Binary connections, Up: Binary connections

6.5.1 特殊值

函数 readBin 和 writeBin 可以传递缺损和特殊值。 在需要修改存储尺寸时，最好不要传输这类特殊值。

R 的逻辑和整型缺损值是 INT_MIN， 它是 C 头文件 limits.h 里面表示 int 最小的值， 通常对应着位模式 0x80000000。

R 的数值和复数类型的特殊值得表示是机器依耐的，还可能是编译器依耐的。 使用它们的最简单的方法是连到一个外部程序，而不是连到用于输出 双精度恒量 NA_REAL，R_PosInf 和 R_NegInf 以及包含 定义了宏 ISNAN 和 R_FINITE 的头文件 Rmath.h 的 Rmath 包。

如果这不能办到，在所有公共的平台上，IEC 60559 （也就是 IEEE 754）算法都可以被采用。因此，标准的 C 工具可以用来测试或设置 值 Inf，-Inf 和 NaN。 在这些平台上，NA 用 NaN 表示，值为 0x7a2 （就是十进制里面的 1954）[13]。

字符缺损值写作 NA，并且没有专门的措施识别 识别作为缺损值的字符值（因为这一步可以在读取它们后 再次赋值的时候判断）。

Next: Reading Excel spreadsheets, Previous: Connections, Up: Top

7 网络接口

- Reading from sockets: 从套接字读取
- Using download.file: 使用 download.file
- DCOM interface: DCOM 接口

- CORBA interface: CORBA 接口

在网络连接的底层水平上交换数据， R 提供的功能非常有限。

Next: Using download.file, Previous: Network interfaces, Up: Network interfaces

7.1 从套接字读取

基本的 R 平台提供一些工具通过 BSD 套接字 让那些支持它们的系统（包括通常的 Linux，Unix 和 Windows 系统上的 R 端口）进行通信。 使用套接字的一个潜在问题是这些工具通常因为安全原因被阻塞 或强迫使用网页缓存。因此，这些功能可能在局域网比外部网有用。 对于新项目，建议使用套接字连接代替直接使用套接字。

早期的底层接口通过函数 `make.socket`， `read.socket`，`write.socket` 和 `close.socket` 来实现的。

Next: DCOM interface, Previous: Reading from sockets, Up: Network interfaces

7.2 使用 `download.file`

函数 `download.file` 通过 FTP 或 HTT 读取来自网络资源的文件，然后 写入到一个文件中。通常这一步可以避免的，因为函数如 `read.table` 和 `scan` 都可以直接从一个 URL 读取内容，它们要么显式地用 `url` 打开一个连接，要么暗含地给 `file` 参数设定一个 URL。

Next: CORBA interface, Previous: Using download.file, Up: Network interfaces

7.3 DCOM 接口

DCOM 是 Windows 用于可能在不同机器上的不同程序间 通讯的一种协议。 从 CRAN 的 Software->Other->Non-standard 下面 可以获得的 Thomas Baier 开发的 `StatConnector` 程序提供了一个连到 R 的 Windows 版本配套的代理 DLL 的接口，同时创建一个 DCOM 服务器。 这可用于和 R 交换简单对象（向量和矩阵） 以及把命令发送给 R。

这个程序有一个 Visual Basic 的演示程序以及 Erich Neuwirth 开发的一个 Excel 插件。这个接口处于另外一个方向和这里考虑的不一样， 即其它应用软件作为客户端（Excel，或用 Visual Basic 写的）， 而 R 作为服务器。

另外一个 (D)COM 服务器可以从 http://www.omegahat.org/ 获得， 这种服务器允许 R 对象以 COM 值输出。那个网站还有包 `RDCOMClient` 和 `SWinTypeLibs`，它们允许 R 作为一个 (D)COM 客户端。

Previous: DCOM interface, Up: Network interfaces

7.4 CORBA 接口

CORBA （通用对象请求代理体系结构， Common Object Request Broker Architecture） 和 DCOM 类似，允许应用程序调用方法或操作，服务器端运行 在其它程序里面的对象。这些应

用程序还可能用不同的语言编写， 还运行在不同的机器上。 Omegahat 项目（http://www.omegahat.org/RSCORBA/）中 有个 **CORBA** 包，目前为 Unix 设计。但是 Windows 版本的也是 有可能设计的。

这个包允许 R 命令用于查找可以获得的 CORBA 服务器 的地址，查询它们提供的方法，并且动态地在这些对象里调用方法。 在这些调用里，作为参数的 R 值在调用时输出，在操作执行时获得。 原始数据类型（向量和列表）默认是输出的，但更为复杂的对象通过 引用导出。这样使用的例子包括和 Gnumeric （http://www.gnumeric.org）电子表格通讯，以及和数据 可视化系统 ggobi 配合使用。

用户可以在 R 里面创建 CORBA 服务器，允许 其它应用程序调用这些方法。例如，用户可能提供对特别的数据集 或一些 R 建模软件的访问。通过联合 R 数据对象和函数， 这些可以动态实现。这样允许用户显式地从 R 里面导出 数据和函数。

用户还可以用 **CORBA** 在 R 里面实现分布式的并行运算。 一个 R 会话作为一个管理器，同时向运行在其它 R 工作会话 上的不同服务器分发任务。这是因为 R 里面实现了 CORBA 调用的异步和后台调用。 更多的信息可以从 Omegahat 项目（http://www.omegahat.org/RSCORBA/）获得。

Next: References, Previous: Network interfaces, Up: Top

8 读取 Excel 电子表格

最常见的 R 数据导入/导出问题可能是`我如何读一个 Excel 电子表格'。 本章汇总了早先给出的建议和可选方案。 注意，大多数建议都是基于 Excel 2007 以前版本的 Excel 电子表格：现在只有一种方法可以读取 .xlsx 格式的文件，就是通过 **RODBC**。

第一个建议就是尽量避免这样做！如果你可以访问 Excel，把你的 Excel 数据 用制表符分隔或逗号分隔的格式导出，然后用 `read.delim` 或 `read.csv` 导入 R。 （在采用逗号作为小数点的欧洲大陆本地系统里面，你可能需要用 `read.delim2` 或 `read.csv2`。） 导出一个 DIF 文件然后用 `read.DIF` 读入是另外一种可能性。

如果你没有 Excel，还有许多其它软件可以用来读这种 电子表格然后导出为文本格式，无论在 Windows 还是 Unix 系统。 例如，Gnumeric （http://www.gnome.org/projects/gnumeric/）和 OpenOffice （http://www.openoffice.org）。你还可以 在这些软件里面展示的电子表格和 R 之间用剪切-粘帖功能： `read.table` 能从 R 控制台读入 或者在 Windows 里面，从 剪贴板（通过 `file = "clipboard"`或 `readClipboard`）。 `read.DIF` 也能从剪贴板读入。

注意，Excel 文件 .xls 不仅仅是一个电子表格：这种文件可能会含有 很多电子表格，而且这些表单能包含公式，宏等等。 不是所有的读者都可以看到第一个表单外的其它表单，并且可能对文件的其它内容比较困惑。

Windows 用户可以用包 **RODBC** 里面的 `odbcConnectExcel` 。 这可以选择 Excel 电子表格中的任何一个电子表格的行和列。 `odbcConnectExcel2007` 可以读 Excel 2007 格式和早期的版本 （假定安装了最新的驱动：见前面的内容）。

也是只用于 Windows 系统，包 **xlsReadWrite** 有一个函数 `read.xls` 来读.xls 文件（基于第三方的 非开源的 Delphi 组分）。

`Perl` 用户捐献过一个模块 `OLE::SpreadSheet::ParseExcel` 和一个程序 `xls2csv.pl` 来 把 Excel 电子表格转换为 CSV 文件。包 **gdata** 在函数 `read.xls` 中队这些模块进行了基本的封装。

Appendix A References

R. A. Becker, J. M. Chambers and A. R. Wilks (1988) *The New S Language. A Programming Environment for Data Analysis and Graphics.* Wadsworth & Brooks/Cole.

J. Bowman, S. Emberson and M. Darnovsky (1996) *The Practical SQL Handbook. Using Structured Query Language.* Addison-Wesley.

J. M. Chambers (1998) *Programming with Data. A Guide to the S Language.* Springer-Verlag.

P. Dubois (2000) *MySQL.* New Riders.

M. Henning and S. Vinoski (1999) *Advanced CORBA Programming with C++.* Addison-Wesley.

K. Kline and D. Kline (2001) *SQL in a Nutshell.* O'Reilly.

B. Momjian (2000) *PostgreSQL: Introduction and Concepts.* Addison-Wesley. Also downloadable at http://www.postgresql.org/docs/awbook.html.

T. M. Therneau and P. M. Grambsch (2000) *Modeling Survival Data. Extending the Cox Model.* Springer-Verlag.

E. J. Yarger, G. Reese and T. King (1999) *MySQL & mSQL*. O'Reilly.

Function and variable index

- `dbSendQuery`: DBI / RMySQL
- `dbWriteTable`: DBI / RMySQL
- `fetch`: DBI / RMySQL
- `file`: Types of connections
- `format`: Export to text files
- `ftable`: Flat contingency tables
- `gzfile`: Types of connections
- `hdf5`: Binary data formats
- `isSeekable`: Listing and manipulating connections
- `make.socket`: Reading from sockets
- `netCDF`: Binary data formats
- `odbcClose`: RODBC
- `odbcConnect`: RODBC
- `odbcConnectDbase`: dBase files (DBF)
- `odbcConnectExcel`: Reading Excel spreadsheets
- `odbcConnectExcel`: RODBC
- `odbcConnectExcel2007`: Reading Excel spreadsheets
- `odbcDriverConnect`: RODBC
- `odbcGetInfo`: RODBC
- `odbcQuery`: RODBC
- `open`: Types of connections
- `pipe`: Types of connections
- `pushBack.`: Pushback
- `pushBackLength`: Pushback
- `read.csv`: Reading Excel spreadsheets
- `read.csv`: Variations on read.table
- `read.csv2`: Variations on read.table
- `read.dbf`: dBase files (DBF)
- `read.delim`: Variations on read.table
- `read.delim`: Reading Excel spreadsheets
- `read.delim2`: Variations on read.table
- `read.DIF`: Reading Excel spreadsheets
- `read.DIF`: Data Interchange Format (DIF)
- `read.dta`: EpiInfo Minitab SAS S-PLUS SPSS Stata Systat
- `read.epiinfo`: EpiInfo Minitab SAS S-PLUS SPSS Stata Systat
- `read.fortran`: Fixed-width-format files
- `read.ftable`: Flat contingency tables
- `read.fwf`: Fixed-width-format files
- `read.mtp`: EpiInfo Minitab SAS S-PLUS SPSS Stata Systat
- `read.octave`: Octave
- `read.S`: EpiInfo Minitab SAS S-PLUS SPSS Stata Systat
- `read.socket`: Reading from sockets
- `read.spss`: EpiInfo Minitab SAS S-PLUS SPSS Stata Systat
- `read.systat`: EpiInfo Minitab SAS S-PLUS SPSS Stata Systat

- `read.table`: Variations on read.table
- `read.table`: Reading Excel spreadsheets
- `read.table`: Input from connections
- `read.xport`: EpiInfo Minitab SAS S-PLUS SPSS Stata Systat
- `readBin`: Binary connections
- `readChar`: Binary connections
- `readClipboard`: Reading Excel spreadsheets
- `readLines`: Using scan directly
- `readLines`: Input from connections
- `reshape`: Re-shaping data
- `scan`: Using scan directly
- `scan`: Input from connections
- `scan`: Imports
- `seek`: Listing and manipulating connections
- `showConnections`: Listing and manipulating connections
- `sink`: Output to connections
- `sink`: Export to text files
- `socketConnection`: Types of connections
- `sqlCopy`: RODBC
- `sqlFetch`: RODBC
- `sqlFetchMore`: RODBC
- `sqlGetResults`: RODBC
- `sqlQuery`: RODBC
- `sqlSave`: RODBC
- `sqlTables`: RODBC
- `stack`: Re-shaping data
- `stderr`: Types of connections
- `stdin`: Types of connections
- `stdout`: Types of connections
- `Sys.localeconv`: Variations on read.table
- `textConnection`: Types of connections
- `truncate`: Listing and manipulating connections
- `unstack.`: Re-shaping data
- `url`: Types of connections
- `write`: Output to connections
- `write`: Export to text files
- `write.csv`: Export to text files
- `write.csv2`: Export to text files
- `write.dbf`: dBase files (DBF)
- `write.dta`: EpiInfo Minitab SAS S-PLUS SPSS Stata Systat
- `write.foreign`: Export to text files
- `write.matrix`: Export to text files
- `write.socket`: Reading from sockets
- `write.table`: Output to connections

Previous: Function and variable index, Up: Top

Concept index

Footnotes

[1] 我不排除有一天人类真的可以造出哆啦 A 梦，但今天 我还是暂时不这样地假设。

[2] 译者注：国外的软件破解版没有我们这么容易方便。还有， 软件太大了，有时，也不愿安装。比如 SPSS，SAS 比 R 大多了。

[3] 译者注：比如德国，1/2=0,5。注意， 他们不用句点表示小数点，他们采用的是逗号。

[4] 译者注：如 SVG。

[5] 通常这一步是很快的，因为 查看第一个条目可以排除大部分可能性。

[6] 译者注： 现在，R 可以对 LC_NUMERIC 重新设置。你可以通过 调用 `Sys.localeconv()` 找到计算机当前使用的设置。

[7] 译者注：可以利用命令生成随机矩阵，

```
    write.table(matrix(rnorm(200*2000), 200), "matrix.dat", row.names=F,
col.names=F)
```

[8] 译者注：很多数据库的新性能或者理念， 大多在 PostgreSQL 上面测试的。

[9] 译者注：DROP 会把一个表彻底删除，而 DELETE 只删除 一个表的内容，会保持表的存在。

[10] 译者注：MySQL 给数据库表命名的时候要小心一点。SQL 标准里面，表的 大小写不敏感，但 MySQL 在 Linux 下面可能大小写敏感。此时可以通过`-O lower_case_table_names=1` 参数启动 mysqld，把表名字统一变成小写。

[11] 译者注：原句为， “*Connections* are used in R in the sense of Chambers (1998), a set of functions to replace the use of file names by a flexible interface to file-like objects.”

[12] 译者注： 就是 eXternal Data Representation，外部数据描述标准

[13] 译者注：原句为“On such platforms `NA` is represented by the `NaN` value with low-word `0x7a2` (1954 in decimal).”