数据分析与决策技术丛书

[PACKT]
PUBLISHING

Mastering Data Analysis with R

R语言数据分析

[美] 盖尔盖伊·道罗齐（Gergely Daróczi） 著

潘怡 译

从数据预处理到数据建模并可视化
使用R语言来解决现实世界中数据科学的难题

机械工业出版社
China Machine Press

数据分析与决策
技术丛书

Mastering Data Analysis with R

R语言数据分析

[美] 盖尔盖伊·道罗齐（Gergely Daróczi） 著

潘怡 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

R 语言数据分析 /（美）盖尔盖伊·道罗齐（Gergely Daróczi）著；潘怡译 . —北京：机械工业出版社，2016.9
（数据分析与决策技术丛书）
书名原文：Mastering Data Analysis with R

ISBN 978-7-111-54795-2

I. R… II. ①盖… ②潘… III. ①程序语言－程序设计 ②数据处理 IV. ① TP312 ② TP274

中国版本图书馆 CIP 数据核字（2016）第 233020 号

本书版权登记号：图字：01-2016-1891

R 语言数据分析

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：何欣阳　　责任校对：董纪丽
印　　刷：北京市荣盛彩色印刷有限公司　　版　　次：2016 年 10 月第 1 版第 1 次印刷
开　　本：186mm × 240mm　1/16　　印　　张：18.25
书　　号：ISBN 978-7-111-54795-2　　定　　价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379642　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzit@hzbook.com

The Translator's Words 译 者 序

R 语言在数据分析与机器学习领域已经成为一款重要的工具，根据 Tiobe、PyPL 以及 Redmonk 等编程语言的人气排名结果显示，它所受到的关注程度正在快速提升，并成为统计领域最具人气的语言选项。了解并掌握 R 语言的编程开发，也就意味着我们能够更高效地分析和处理数据。

位于英国伯明翰的 Packt 公司是世界上发展最快、产品最丰富的技术书籍出版商之一，本书是 Packt 公司近年推出的又一本技术力作，全书一共分为 14 章，重点探讨了数据预处理的方法，包括数据获取、筛选、重构、建模、平滑以及降维，本书还介绍了分类和聚类等几种主要的数据分析方法，以及网络数据、时序数据、空间数据及社交媒体数据等一些特殊类型数据的分析处理。本书以数据科学家、R 开发人员和具备基础 R 语言知识的工程师为目标读者，通过阅读本书，读者能够更多地了解有关 R 的高级功能及工具，同时提高 R 语言的开发能力。为了照顾初学者，尽管作者没有过多地介绍 R 语言的基本知识，但依然很贴心地提供了相关的参考资料，以帮助读者快速进入角色并掌握相关技术。

本书作者盖尔盖伊 • 道罗齐（Gergely Daróczi）是一位狂热的 R 用户及开发人员，也是 rapporter.net 网站的创始人及 CTO，现就职于洛杉矶的 www.card.com 网站，担任首席 R 语言开发及研究的数据专家。受作者自身的研究背景影响，本书花了相当多的篇幅分析在数据预处理环节开发人员可能遇到的各类问题，并给出了多种经过实践证明的解决方案。书中所有源代码和实验数据在华章网站（www.hzbook.com）上都可以免费下载，相信阅读完本书并亲自动手实践完成所有案例方法后，读者将对在数据科学领域应用 R 语言有更深入的了解，也将对数据处理及分析有更多的领悟及体会。

本书能够得以出版，要感谢机械工业出版社的缪杰和何欣阳编辑，他们在翻译过程中给予了我很多建设性的指导意见。其次，还要感谢吴怡编辑，是她让我与机械工业出版社结缘。

由于教学科研需要，译者很早就已经接触了R语言，之前翻译《机器学习与R语言实战》一书，也让我获益匪浅，但由于学科发展速度日新月异，在翻译过程中我仍然遇到了一些问题，尽管在此期间我查阅了大量的文献及网络资源，并逐字逐句地对译稿进行了反复推敲和琢磨，但还是不可避免地存在错误和疏漏之处，还望各位读者不吝指正。

潘怡

2016年8月

Preface 前　言

自 20 多年前发源于学术界以来，R 语言已经成为统计分析的通用语言，活跃于众多产业领域。目前，越来越多的商业项目开始使用 R，兼之 R 用户开发了数以千计易于上手的开发包，都使得 R 成为数据分析工程师及科学家最常用的工具。

本书将帮助读者熟悉 R 语言这一开源生态系统，并介绍一些基本的统计背景知识，以及一小部分相关的数学知识。我们将着重探讨使用 R 语言解决实际的问题。

由于数据科学家在数据的采集、清洗及重构上将耗费大量时间，因此本书首先将通过第一手实例来重点探讨从文件、数据库以及在线资源中导入数据的方法，然后再介绍数据的重构和清洗——不包含实际的数据分析，最后几章将对一些特殊的数据类型以及经典的统计模型和部分机器学习算法进行说明。

本书主要内容

第 1 章从与所有数据相关项目都有关的关键性的第一步——从文本文件和数据库中导入数据开始。重点探讨使用优化的 CSV 分析器把数据载入 R，预筛选数据，并对不同数据库后台对 R 的支持能力进行比较。

第 2 章介绍如何使用面向 Web 服务和 API 通信的包实现数据的导入，包括如何从主页上整理和抽取数据。还将对处理 XML 和 JSON 格式数据进行概括性说明。

第 3 章继续介绍基础的数据处理知识，包括多种数据筛选和聚集，并对 data.table 和 dplyr 这两个常见开发包在性能和使用语法方面进行比较。

第 4 章介绍更多有关复杂数据类型的转换方法，相关函数包括处理数据子集、数据合并、长宽表数据格式到适合用户需要的工作流源数据格式之间的转换等。

第 5 章开始介绍真实的统计模型，包括回归的概念、常用回归模型等。这一章篇幅不长，还介绍了模型测试的方法以及基于真实数据集如何解释某个多元线性回归模型结果。

第 6 章在前述章节的基础上，探讨了预测变量的非线性关联，以及诸如逻辑回归和泊松回归等广义线性模型的样例。

第 7 章介绍一些新的非结构化数据类型，读者将通过实践文本挖掘算法及对结果的可视化处理，了解使用统计模型来处理类似这样一些非结构化数据的方法。

第 8 章探讨有关原始数据集的另一个常见问题。大多数时候，数据科学家需要处理脏数据，包括去掉错误数据、孤立点以及其他不正确的值，同时又要将缺失值带来的影响降到最低。

第 9 章介绍如何从大数据中进行特征提取，假设我们已经装载了一个干净的数据集，并且完成了格式转换，当我们开始处理高维变量时，需要采用一些统计方法来进行降维以及其他包括主成分分析、因子分析和多维尺度分析等方法完成连续变量的转换。

第 10 章讨论使用监督及非监督统计和机器学习方法来处理样本分组问题。这些方法包括层次聚类、k 均值聚类、潜类别模型、判别分析、逻辑回归和 k 近邻算法，以及分类树和回归树。

第 11 章重点探讨一类特殊的数据结构，包括其基本概念以及可视化网络分析技术，igraph 包是该章的重点。

第 12 章展示如何通过平滑、季节性分解以及 ARIMA 等方法处理分析时间 – 日期数据及其相关值，同时还将讨论有关预测和孤立点检测等技术。

第 13 章探讨一类重要的数据维度——空间维，重点会放在通过主题图、交互图、等高线和冯洛诺伊图完成空间数据的可视化。

第 14 章提供了一个更完整的样例，该样例中包含了很多前述章节中提到的方法来帮助读者复习这本书所学习到的主要内容，以及应对未来工作中可能遇到的问题和困难。

附录给出了 R 语言的帮助索引，以及对前述章节中涉及内容的补充阅读。

阅读准备

本书所展示的代码都应该在 R 控制台内运行，读者需要事先安装好 R，可以从 http://r-project.org 下载免费软件以及为所有主流操作系统准备的安装指南。

本书并不会探讨其他更深入的内容，例如在集成开发环境（Integrated Development Environment IDE）下使用 R 的方法，尽管 IDE 为诸如 Emacs、Eclipse、vi、NotePad++ 都提供了非常棒的插件和扩展。当然，我们还是建议读者能够使用 RStudio，这是一个为 R 开发的开源免费 IDE，访问地址为 https://www.rstudio.com/products/RStudio。

除了基础的 R 包，我们还会使用到部分用户自己提供的 R 包，它们大多都可以很容易

地从 R 综合典藏网（Comprehensive R Archive Network，CRAN）处下载安装。附录中列出了本书用到的开发包以及多个版本。

如果要从 CRAN 安装包，读者要确保网络通畅。假如要下载二进制文件，可以在 R 控制台调用 install.packages 命令：

```
> install.packages('pander')
```

本书中所提到的部分包在 CRAN 上下载不了，但也许可以从 Bitbucket 或者 GitHub 处找到安装文件，然后再通过调用 devtools 包的 install_bitbucket 和 install_github 函数完成安装。Windows 用户则需首先从 https://cran.r-project.org/bin/windows/Rtools 处安装 rtools 包。

安装完毕后，我们应该在使用包之前先将其装载到 R 会话中，附录中列出了所有包的目录，而每一章的一开始则对相关的源码和 R 命令做了介绍：

```
> library(pander)
```

我们极力建议读者下载安装本书的样例源码（可以参考前言的“样例源码下载”小节），这样读者就可以在 R 控制台很容易地复制和粘贴相关命令，而不需要再按照书中文字输入代码。

如果读者之前没用过 R 语言，最好能够先从 R 主页上阅读一些免费的介绍性文章和帮助手册，本书附录中也列出了一些推荐阅读材料。

读者人群

如果你是数据科学家或者是 R 开发人员，希望更多地了解有关 R 的高级功能及工具，那么这本书就是为你而写。本书希望读者已经具备基础的 R 语言知识，了解数据库的逻辑。如果你是数据科学家、工程师或分析师，希望提高自己对 R 语言的开发能力，那么这本书也适合你。尽管需要掌握一些基本的 R 知识，本书还是为你提供了相关参考文档，能够帮助你快速进入角色并掌握相关技术。

本书约定

本书中任何将在 R 控制台输入或输出的命令行将采用如下格式：

```
> set.seed(42)
> data.frame(
+   A = runif(2),
+   B = sample(letters, 2))
          A B
1 0.9148060 h
2 0.9370754 u
```

符号“>”有提示的意思，指此处 R 控制台正在等待要输入执行的命令。如果命令长度超过一行，则第一行还是用“>”开头，但剩下的其余行都要在行首添加符号“+”，代表该行不是一个完整的命令（例如，缺圆括号或引号）。命令的输出不需要增加任何首字母，字体采用和输入文本相同的等宽字体。

新出现的术语和重要的文字将用粗体表示。

警告或重要提示将跟在这样的符号后面。

小窍门或诀窍将跟在这样的符号后面。

样例源码下载

你可以从 http://www.packtpub.com 通过个人账号下载你所购买书籍的样例源码。如果你是从其他途径购买的，可以访问 http://www.packtpub.com/support，完成账号注册，就可以直接通过邮件方式获得相关文件。

你也可以访问华章图书官网：http://www.hzbook.com，通过注册并登录个人账号，下载本书的源代码。

下载书中彩图

我们还为读者准备了一个 PDF 文件，该文件包含了本书所有截图和样图，可以更好地帮助读者理解输出的变化。你可以从以下地址下载：

http://www.packtpub.com/sites/default/files/downloads/1234OT_ColorImages.pdf

Contents 目　录

第 1 章 Chapter 1

你好，数据！

大多数 R 项目都必须从数据导入到 R 的会话中开始，由于 R 语言能够支持多种文件格式和数据库后台，因此可以使用相当多的数据导入方法。本章，我们不会再讨论基础的数据结构，因为你应该已经对它们非常熟悉了。本章的重点将放在大数据集的导入以及处理一些特殊的文件类型。

如果读者希望对标准工具做一个粗略的回顾，复习一下普通类型数据导入的方法，可以参考官方有关 CRAN 介绍的手册，地址为：http://cran.r-project.org/doc/manuals/R-intro.html#Reading-data-from-files，或者访问 Rob Kabacoff 的 Quick-R 站点：http://www.statmethods.net/input/importingdata.html，该网站总结了大多数 R 任务中将使用的关键字和提示信息列表，更多相关内容，请参考本书附录。

尽管 R 语言拥有其自己的（序列化）二进制 RData 及 rds 文件格式类型，这种文件格式也可以非常方便地被 R 用户用来存放 R 对象的元数据信息。但大多数时候，我们还是需要能够处理一些由我们的客户或老板要求使用的其他类型数据。

平面文件是这其中最常见的一类数据文件，在这样的文件中，数据存放在简单的文本文件中，数据值之间通常会以空格、逗号，或者更常见的分号隔开。本章将对 R 语言提供的几种用于装载这些类型文档的方法展开讨论，并就哪种方法最适合于导入大数据集进行测试。

某些时候，我们也可能仅对一个数据集的子集感兴趣，并不需要对整个数据集进行处理。由于数据存放在数据库时都是以结构化的方式进行预处理的，因此，我们可以只使用简单并且有效的命令就可以查询得到我们需要的子集。本章 1.4 节将着重探讨三类最常用的数据库系统（MySQL、PostgreSQL 和 Oracle）与 R 进行交互的方法。

除了对部分常用工具以及其他一些数据库后台进行一个简要说明外，本章还将展示如何将 Excel 电子表格导入到 R 中，这种导入并不需要事先将电子表格文件转换为 Excel 文本文件或 Open/LibreOffice 格式文件。

当然，本章要讨论的内容绝不仅仅局限于文件格式、数据库连接以及类似一些让人提不起兴趣的内容。不过，请记住数据分析工程师总是首先从导入数据起步，这一部分的工作是不可回避的，必须要保证我们的机器和统计环境在进行实际的分析之前首先先弄清楚数据的结构。

1.1 导入一个大小合适的文本文件

本章的标题也可以换成“你好，大数据！”因为本章主要探讨如何将大数据装载到 R 会话中。但是，到底什么是大数据呢？究竟在 R 中处理多大规模的数据量会比较困难呢？合适的规模怎么定义呢？

R 原本是为处理单机规模的数据而设计的，因此比较适合数据集规模小于实际可用的 RAM 大小的情况，但要注意有时候我们必须考虑在做一些计算操作时，程序对内存的需求会增加，例如主成分分析。在本节中，将这类规模的数据集称为大小合适的数据集。

在 R 中完成从文本导入数据的操作非常简单，可以调用 read.table 函数来处理任何规模合适的数据集，唯一要考虑的就是数据读写所需的时间。例如，25 万行的数据集？可以参见：

```
> library('hflights')
> write.csv(hflights, 'hflights.csv', row.names = FALSE)
```

注意，我们对本书所有的 R 命令及其输出都采用特殊格式的文本显示。其中，R 命令以符号“>”开始，属于同一命令的不同行之间以“+”连接，与 R 控制台的处理方式类似。

没错，我们刚刚从 hflights 包中将 18.5MB 大小的文本文件下载到硬盘上，该文件包括了 2011 年从休斯顿（Houston）起飞的航班的部分数据：

```
> str(hflights)
'data.frame':  227496 obs. of  21 variables:
 $ Year             : int  2011 2011 2011 2011 2011 2011 2011 ...
 $ Month            : int  1 1 1 1 1 1 1 1 1 1 ...
 $ DayofMonth       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ DayOfWeek        : int  6 7 1 2 3 4 5 6 7 1 ...
 $ DepTime          : int  1400 1401 1352 1403 1405 1359 1359 ...
 $ ArrTime          : int  1500 1501 1502 1513 1507 1503 1509 ...
 $ UniqueCarrier    : chr  "AA" "AA" "AA" "AA" ...
 $ FlightNum        : int  428 428 428 428 428 428 428 428 428 ...
 $ TailNum          : chr  "N576AA" "N557AA" "N541AA" "N403AA" ...
 $ ActualElapsedTime: int  60 60 70 70 62 64 70 59 71 70 ...
 $ AirTime          : int  40 45 48 39 44 45 43 40 41 45 ...
```

```
$ ArrDelay          : int  -10 -9 -8 3 -3 -7 -1 -16 44 43 ...
$ DepDelay          : int  0 1 -8 3 5 -1 -1 -5 43 43 ...
$ Origin            : chr  "IAH" "IAH" "IAH" "IAH" ...
$ Dest              : chr  "DFW" "DFW" "DFW" "DFW" ...
$ Distance          : int  224 224 224 224 224 224 224 224 224 ...
$ TaxiIn            : int  7 6 5 9 9 6 12 7 8 6 ...
$ TaxiOut           : int  13 9 17 22 9 13 15 12 22 19 ...
$ Cancelled         : int  0 0 0 0 0 0 0 0 0 0 ...
$ CancellationCode  : chr  "" "" "" "" ...
$ Diverted          : int  0 0 0 0 0 0 0 0 0 0 ...
```

用 hflight 包我们能非常方便地处理海量航线数据的子集，该数据集源自美国交通统计局的研究和创新技术局提供的海量航班数据集的子集，原始数据集中包括了自 1987 年以来，所有 US 航班的计划及实际出发 / 到达时间和其他一些我们可能感兴趣的信息。该数据集经常被用于验证机器学习及大数据技术。更多有关该数据集的详细内容，可以参考以下网址来获得有关列的描述以及其他元数据的内容：http://www.transtats.bts.gov/DatabaseInfo.asp?DB_ID=120&Link=0.

我们将使用这个包括了 21 列数据的数据集作为数据导入的测试平台。例如，使用 read.csv 测试导入 CSV 文件的时间。

```
> system.time(read.csv('hflights.csv'))
   user  system elapsed
  1.730   0.007   1.738
```

从某个 SSD 站点下载这些数据大约需要 1.5 秒，相对来说耗时还算可以接受。我们可以指定列数据的转换类型而不采用默认的 type.convert（参见 read.table 的文档获得更多详细信息，在 SatckOverflow 的搜索结果也表明有关 read.csv 的问题看起来是大家都很关心也经常提问的内容）来提高速度。

```
> colClasses <- sapply(hflights, class)
> system.time(read.csv('hflights.csv', colClasses = colClasses))
   user  system elapsed
  1.093   0.000   1.092
```

这个结果已经好了很多！但它可信吗？在使用 R 语言掌握数据分析的道路上，我们还将实践更多可靠的测试——对同一任务重复 *n* 次测试，然后再对仿真结果进行汇总。通过这个方法，我们可以得到关于数据的多种观测结果，并将它们用于分析确定结果中的统计的显著差异。microbenchmark 包就为类似任务提供了一个非常好的框架：

```
> library(microbenchmark)
> f <- function() read.csv('hflights.csv')
> g <- function() read.csv('hflights.csv', colClasses = colClasses,
+                          nrows = 227496, comment.char = '')
```

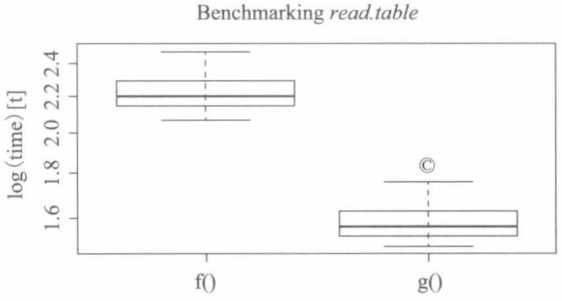
```
> res <- microbenchmark(f(), g())
> res
Unit: milliseconds
 expr       min        lq    median        uq       max neval
  f() 1552.3383 1617.8611 1646.524 1708.393 2185.565   100
  g()  928.2675  957.3842  989.467 1044.571 1284.351   100
```

我们定义了两个函数：函数 f 为 read.csv 的默认设置，在函数 g 中，我们对之前两列数据类型进行了更新以提高执行效率。其中，参数 comment.char 将通知 R 不需要在被导入的文件中寻找注释，参数 comment.char 确定了从文件中导入的行数，以节约导入操作所需的部分时间和空间。将 stringAsFactors 设置为 FALSE 也可以提高一点文件导入速度。

使用一些第三方工具可以确定要导入的文本文件的行数，例如 Unix 上的 wc，或使用 R.utils 包中自带的 countLines 函数，不过后者速度要稍微慢一点。

回到对结果的分析中，我们可以在图形中来展现中位数以及一些其他相关统计值，这些结果都是默认运行 100 次所得：

```
> boxplot(res, xlab = '',
+   main = expression(paste('Benchmarking ', italic('read.table'))))
```



两者之间的差异看起来非常明显（读者也可以通过其他一些统计实验来验证这个结果），仅通过 read.table 函数的参数调优，我们就将性能提高了 50% 以上。

规模大于物理内存的数据集

如果从 CSV 文件中导入的数据集大小超过了机器的物理内存，可以调用一些专为这类应用而设计的用户开发包。例如，sqldf 包和 ff 包都支持基于特定数据类型以 chunk 到 chunk 方式装载数据集。前者使用 SQLite 或者类似 SQL 的数据库后台，而后者则使用与 ffdf 类对应的数据框将数据存储到硬盘上。bigmemory 包也提供了类似的功能。稍后将介绍相关的样例（可用于测试）：

```
> library(sqldf)
> system.time(read.csv.sql('hflights.csv'))
   user  system elapsed
```