

The Little SAS Book 中文版

[美]Lora D. Delwiche (洛拉 · D. 德尔维奇)
Susan J. Slaughter (苏珊 · J. 斯劳特) 著
小小SAS翻译组 译

清华大学出版社
北京

The correct bibliographic citation for this manual is as follows: Delwiche, Lora D., and Susan J. Slaughter. 2012. *The Little SAS® Book: A Primer, Fifth Edition*. Cary, NC: SAS Institute Inc.

The Little SAS® Book: A Primer, Fifth Edition

Copyright © 2012, SAS Institute Inc., Cary, NC, USA

EISBN 978-1-61290-343-9 (Hard copy)

北京市版权局著作权合同登记号 图字：01-2017-7185

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。

版权所有，侵权必究。侵权举报电话：010-62782989 13701121933

图书在版编目(CIP)数据

The Little SAS Book中文版 / (美)洛拉 · D. 德尔维奇(Lora D. Delwiche), (美)苏珊 · J. 斯劳特(Susan J. Slaughter)著；小小SAS翻译组译. — 北京：清华大学出版社，2018

书名原文：The Little SAS Book: A Primer, Fifth Edition

ISBN 978-7-302-48710-4

I . ①T… II . ①洛… ②苏… ③小… III. ①统计分析—应用软件 IV. ①C819

中国版本图书馆 CIP 数据核字(2017)第 277719 号

责任编辑：刘 洋

封面设计：李召霞

版式设计：方加青

责任校对：王荣静

责任印制：王静怡

出版发行：清华大学出版社

网 址：<http://www.tup.com.cn>, <http://www.wqbook.com>

地 址：北京清华大学学研大厦 A 座 邮 编：100084

社 总 机：010-62770175 邮 购：010-62786544

投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈：010-62772015, zhiliang@tup.tsinghua.edu.cn

印 装 者：三河市君旺印务有限公司

经 销：全国新华书店

开 本：187mm×235mm **印 张：**26 **字 数：**496 千字

版 次：2018 年 2 月第 1 版 **印 次：**2018 年 2 月第 1 次印刷

印 数：1 ~ 5000

定 价：99.00 元

产品编号：076361-01

《The Little SAS Book 中文版》

编 委 会

翻 译：小小 SAS 翻译组

谷鸿秋，国家神经系统疾病临床医学研究中心
刘欣宇，赛仕软件（北京）有限公司
孔 雁，赛仕软件（北京）有限公司
李 默，赛仕软件研究开发（北京）有限公司
辛 岩，广州佰聆数据股份有限公司
李兆钧，赛仕软件（北京）有限公司
杨志欣，赛仕软件（北京）有限公司
钱明凤，赛仕软件（北京）有限公司
王 伟，赛仕软件研究开发（北京）有限公司
杨 珂，赛仕软件（北京）有限公司

审 核：

马晓丽，赛仕软件研究开发（北京）有限公司
王 伟，赛仕软件研究开发（北京）有限公司
薛剑平，赛仕软件研究开发（北京）有限公司
栾思凯，赛仕软件研究开发（北京）有限公司
曾秋媚，赛仕软件（北京）有限公司

项目管理：蒋顺利，北京仁科互动网络技术有限公司

责任编辑：刘 洋，清华大学出版社



《The Little SAS Book 中文版》

各界联袂推荐

数据科学的发展日新月异，新成果层出不穷。作为数据分析领域的领导者，SAS 也经历了 40 余年的变革，从早期的大型机软件，到今天开放的 Viya 云端分析平台。SAS 在企业的核心业务领域里继续扮演着重要的角色，SAS 的学习者与日俱增。诞生于 1995 年的 *The Little SAS Book* 一直以来是 SAS 学习领域里的翘楚，是当仁不让的第一推荐用书。幸运的是，今天它终于跟我们广大的中国读者见面了。SAS 公司 CEO 吉姆·古德奈特博士和原著者都为中文版的出版特别撰写序言，可见这本书的分量。祝大家能够从中找到学习 SAS 的乐趣。

—— 刘政
SAS 中国研发中心总经理

一本好的入门教材，既能很快带领读者进入新的领域，又能激发读者对该领域的兴趣。*The Little SAS Book* 经洛拉·D. 德尔维奇和苏珊·J. 斯劳特两位女士 20 多年潜心耕耘，不断锤炼，数易其版，日臻完善，堪称 SAS 入门之经典力作。其结构合理严谨，编排独特有序，内容由浅入深、循序渐进、丰富实用，备受 SAS 爱好者推崇，读者无不从中获益。小小 SAS 翻译组推出的《The Little SAS Book 中文版》，追求“信、达、雅”之境界，精心打磨，反复推敲，精益求精。无论是初学者还是专业统计人士都值得一读！

—— 陈峰
南京医科大学生物统计教授

风险是经济社会的固有基因，金融风险管理则是优化金融结构的基因工程。SAS 软件通过大数据统计分析和数据挖掘等技术，为金融风险识别、监测和控制提供了有效的

工具。*The Little SAS Book* 以初学者的视角，浅显易懂的说明，简单直观的例子，向读者开启了功能强大且操作简洁的 SAS 之门。《The Little SAS Book 中文版》在精确表达原书意思的同时，做到了语言流畅，行文优雅。相信一定会成为中国 SAS 爱好者的必备入门书目！

—— 陈忠阳

中国人民大学财政金融学院金融风险管理教授

在“互联网+”时代，以移动互联网、大数据、人工智能为代表的新兴技术不断推动人类社会高速发展，如果说 Google 借助搜索引擎的海量数据预测出流感病毒爆发，向人们展现了大数据技术的神奇，那么 AlphaGo 大战李世石则让人们看到了大数据技术的力量。我所在的银行业已逐渐从传统的依赖线下网点的人力资本经营模式，转变为依托线上创新变革的技术资本经营模式：从早期的产品业务驱动发展模式，逐渐向数据技术驱动发展模式转变。

借用狄更斯的话：这是最好的时代，这是最坏的时代。对于大数据而言，这是最缺数据的时代，这是最不缺数据的时代。互联网技术创造了海量的数据，却又将之分离。一家公司是否具备强大的技术能力充分利用超高维度稀疏数据，决定了它能否在大数据时代的数据海洋扬帆远航。SAS 作为数据分析领域的常青树，将在大数据时代最大化发挥其价值。SAS 的解决方案几乎覆盖了金融业的客户、营销、财务、风险管理等业务领域的数据分析工作。

本书作为 *The Little SAS Book* 的中文译本，它很好地继承了原版书籍的专业性及实用性，相信会让广大中国读者受益匪浅，也会进一步提升 SAS 在中国数据分析领域的影响力和贡献。

—— 孙中东

上海华瑞银行副行长兼首席信息官

技术的进步正在快速影响全球各产业的经营模式与运营管理，更让产业间的界线愈来愈模糊。技术遇见不同的领域，碰撞出大小不一的火花，加速许多产业的发展，譬如：技术遇见信息，成为 IT (Information Technology)；技术遇见数据，成为 DT (Data Technology)；技术遇见金融，成为 FinTech (Financial Technology)。

在技术的协助下，数据分析可以运用相较以往更有效的方式，以更快的速度 (Velocity)

处理海量（Volume）、多样（Variety）的大数据，提升拟定战略与制订计划的实际成效。简要来说，这个世界原本就充满着各种数据，数据背后连接着各式各样的逻辑，通过数据技术，让我们能够看清楚这个世界真正的轮廓，掌握事物运行背后的关键。

数据技术要达成的目标是什么？以玉山银行投入数据分析 13 年的经验，我们认为目标就是要创造人的价值，打造更美好的生活。如果只是将精准的数据做理性的分析，没有应用在与人相关的事物上，不能提升人民的生活水平，这样的数据是没有影响力的。唯有以人为本，以数据为媒介，以体验为核心，才能发挥数据的最大效用，创造美好的消费者旅程，为人民创造价值。

以人为本的实现方式就是找出消费者的 Jobs to be done，Jobs 为消费者在特定情况下想要解决的问题，通过了解消费者真正的需求以及各种决策背后的逻辑，提出最适合的产品及服务，并且持续地调整与精进，才能提供良好的体验，这样的方式也是创新的精髓。

优秀的数据科学家会以开放的胸怀接受各种新事物的挑战，融合专业知识与数据技术，学习各产业的实务经验，避开相关性的陷阱，挖掘出数据间的因果关系，揭开数据背后的奥秘。因此，深耕技术与跨界学习将是数据科学家制胜的最重要因素。*The Little SAS Book* 作为学习数据技术的经典读物，通过浅显易懂的文字与图表、具体的案例及详细的步骤，正成为帮助大家学习和掌握数据分析、获得成功的优质工具书。

最后，期望通过大家共同的努力，我们不仅能够逍遥自在地徜徉在浩瀚的数据世界里，更能够一起创造更美好的未来。

—— 李正国

中国台湾玉山银行数位金融长，《亚洲银行家》2017 年度台湾创新领袖奖获得者

在大数据时代，数据分析是驱动决策和优化的代名词。SAS 是全球数据分析软件市场的领导者，无论您是否渴望成为 SAS 专业人士或了解数据分析科学，我强烈推荐阅读这本经典的 *The Little SAS Book* 的中文版。相信你阅读之后一定有所收获。

—— 梁建文

中信银行（国际）有限公司资讯科技及营运总监

我在美国的金融和保险业从事数据分析、建模及与大数据相关的工作达 20 多年，*The Little SAS Book* 这本书对我最初入门学习和掌握 SAS 起到了非常关键的作用。这是一本非常方便好用的 SAS 入门书，尤其适合于初学者，是灵活掌握和应用 SAS 必不可少的

一本参考书。此书浅显易懂，简单实用，很易于查找答案，是在美国 SAS 用户世界里面广泛流传的一本深受大家喜爱的书。我相信国内的 SAS 精英们也一定会喜欢这本书。

—— 来凤嬅
美国银行信用卡风险管理战略部副总裁

我是 1999 年第一次接触 SAS 软件，当时觉得很神奇，到目前还是印象很深刻。统计是一门基于数据，对社会现象或自然现象进行研究分析的方法，是探索未知世界的重要工具，其前提是需要有大量的数据、分析思路和分析工具。SAS 已经将我们几乎能够想象的分析方法、思路和过程整合到一起。2007 年，我们华安基金指数与量化部门购买了 SAS 的服务器版本，一直使用到现在，已经成为我们构建金融数据平台、沟通量化投资模型的重要平台和工具，也成为量化投资的管理平台。*The Little SAS Book* 作为我很多同事使用 SAS 的入门参考、手头必备书籍，比较直观、清晰地给出了 SAS 最为精华的部分，使得 SAS 的学习和应用更为便捷和富有乐趣。作为国内规模最大的指数与量化投资团队之一的负责人，我愿意推荐大家使用这本经典读物的中文版，在 SAS 的平台上实现自己的梦想。

—— 许之彦
华安基金管理有限公司指数与量化事业部总经理、基金经理

我接触 SAS 数据分析软件是在 2005 年年初，当时上汽通用汽车有限公司已拥有上海金桥、山东烟台与辽宁沈阳三个生产制造基地，在国内市场保有量已达上百万辆，上市新车逐年提升，年销量迅速达至几十万乃至上百万辆之多，怎么确保对销售的车辆在使用过程中可能产生的批量问题进行预警摆在了我们 IT 人面前，需要我们思考。这时，SAS 数据分析软件映入了我们的眼帘，上千的数学算法与分析模型让我们开了眼界。当年，经过 4 个月的项目实施，原本需半年时间的数据收集才能发现市场产品批量问题，我们只用了一周时间，并及时反馈至相关部门，把产品质量问题及时修复，提高了我们的产品质量，提升了客户的满意度和忠诚度。我们项目当年的投资回报率高达 1200%，取得了非常好的经济效益。

当今世界的万物互联、大数据、云计算、VR、AR、AI、深度学习，都离不开数据分析，离不开分析模型，离不开算法。《The Little SAS Book 中文版》这本书为我们想入门或想了解这领域知识的人们提供了一个非常好的捷径。如果你认真阅读了这本书，相信你

一定会有收获。

—— 王健
上汽通用汽车有限公司首席架构师

The Little SAS Book 是本小而精的经典 SAS 入门工具书，基本是 SAS 初学者的必备。此书通俗易懂，从介绍 SAS 软件开始，细致而翔实地覆盖了数据分析的关键步骤，包括数据输入、数据转换、数据汇总、数据存储、总结分析结果及报表等，同时还包括了 SAS 编程中出现问题时如何解决。《The Little SAS Book 中文版》是初学者的好伙伴，也是初学者的导师。

随着大数据时代的到来，大数据分析和应用更是刻不容缓。智能分析、机器学习和深度学习已经被广泛应用到各个领域，帮助社会各个领域进行有分析驱动的智能化管理和智慧决策。SAS 公司作为数据分析领域的全球领袖，40 多年来一直专注于数据分析的研究和创新，SAS 编程专家们也不断地对 *The Little SAS Book* 进行着完善，与时共进，更好地为 SAS 爱好者服务。

此次中文版的问世，是对中国 SAS 粉丝的一份厚爱。中科聚信（SCAI）作为国内分析智能和金融科技领域的领袖企业，同时也是 SAS 在中国的唯一金牌合作伙伴，公司员工很多是 SAS 的忠实爱好者，因此对于此中文版的问世也是非常兴奋和开心。“千里之行，始于足下，”《The Little SAS Book 中文版》将是我们在数据分析的海洋里乘风破浪的好伙伴。

—— 马占军
中科聚信信息技术（北京）有限公司董事长兼 CEO

SAS 于我，亦师、亦友，有情、有忆！

仍然记得 9 年前，我刚加入 SAS 中国区从事管理工作的时候，虽是历经十数年 IT 沙场的老将，也曾趟过硬件、软件（中间件、BI）的不同 IT 时代的河流，仍然还是不小心被 DT 的不同闪了一个趔趄，且不说 DT 与 IT 完全不同的商业模式所带来的水土不服，彼时的中国市场对于数据应用的重视尚属拂晓前的沉寂。

“忽如一夜春风来，千树万树梨花开”，这几年来随着国内经济的快速发展以及各级政府从上至下的重视，数据应用的春天如约而至，大数据已成为当下最热的风口之一，我本人也响应李总理“创新创业”的号召，创办了自己的企业，专注于大数据的

挖掘分析应用，帮助客户提炼数据的巨大价值。

SAS 的分析工具作为全球数据分析领域的王者，而本书作为全球 SAS 使用者的经典工具教材，相信这次中译版的出版发行，将给广大中国用户带来方便和帮助。很高兴佰聆数据的员工辛岩作为国内 SAS 用户的佼佼者，也参与了本书的翻译工作。

“路漫漫其修远兮”，中国的数据应用之路刚解缆起航，未来仍然任重道远，让我们一起努力，尽快完成这个领域又一次从追赶到超越的历程！幸哉快哉！

—— 杨钊

广州佰聆数据股份有限公司董事长兼总经理

我从 2003 年开始在大学教数据分析课程，用的程序语言就是 SAS。十多年来，也培养了近千名 SAS 程序员。不过当时一直苦于没有很好的 SAS 中文教材，今天看到由论坛版主谷鸿秋博士参与翻译的此书，格外欣喜。该书浅显易懂，又简明扼要，经典之余仍不失风趣，生动的图示给本书加分不少，严谨的写作态度继承了 SAS 公司的一贯作风，是学习 SAS 的一本很好的参考书！相信《The Little SAS Book 中文版》能成为学习数据分析、SAS 语言、大数据读者的又一优良选择，也希望广大坛友能借此书进入 SAS 的数据分析世界！

—— 赵坚毅

经管之家（原人大经济论坛）创始人

我是从初学 SAS 就开始看 *The Little SAS Book* 这本书的，当时是怀着对 SAS 的敬畏之心开始学习的，觉得这应该是一种非常难以学习的语言（看到网上都是这么说的），我原定的一个月学习时间，最后实际只用了不到一个星期。可能是因为我有一定的编程基础，但是我觉得更多的是 SAS 的语言结构非常直接地刺痛了我的心，这原本就是很多做编程的人需要的语言形式！

这本书没有一上来去举一些高大上的例子来炫耀语言的功能，也没有给我们去展示 SAS 的强大，而是耐心地告诉我们 SAS 为一个编程的人，一个做数据分析的人考虑了什么。书中一上来就告诉我们 SAS 的语言结构是如此的自由、不拘一格，这恰恰是一个编程人所追求的。严格的语言结构虽然看起来完美，但是那不是我们真正想要的。在互联网飞速发展的今天，自由才是真正的方向，这让我们怎么能不喜欢 SAS !

有人说 SAS 是用来做数据分析的，但是我用过之后，印象更加深刻的是 SAS 的数

据管理功能。这本书的作者也深知这一点，书中花了大量的篇幅来告诉我们如何把数据导入 SAS 当中，如何让数据以我们想要的样子展现出来，初识数据分析的人对于这些部分可能并不感兴趣，他们可能期待的是高大上的数据分析方法如何实现，但是当他们真正开始进入数据分析，他们就会马上知道其实数据导入这些工作最耗费时间，最容易出错。*The Little SAS Book* 给我们的功能才是最能帮助到我们的功能。这本书的内容对于数据分析的老手同样有益，这些细致的输入输出讲解可以让我们效率提升不少。我是在高校中开过几年 SAS 数据分析课程的，一开始我认为学生用 SAS，关键是要会实现一些统计分析方法，和先修的多元统计、时间序列等课程相结合。到第一批学生毕业答辩的时候我才发现，很多同学虽然学了四年的统计专业，但连基本的分类汇总、多表拼接、报表生成都不会，他们知道图可以表现出数据的特点，但是他们根本不知道应该画什么图，应该针对什么去画图。这或许可以归咎于课程体系的设计，但是更多的我觉得是我们缺少对于数据的理解，从老师到学生都觉得学一些高大上的方法是有益的，殊不知这些方法是要首先有一个完美的数据集。数据集从哪里来？课本可以给我们提供完美的实例，但是真正到了数据分析的时候，完美的数据集本身是不存在的，需要我们花时间、花精力去构建，老师或许不愿意花时间讲这些内容，但是学生真的应该好好去学习。本书应该可以教你这些，它的语言浅显易懂，配图细致直观，可以很好地帮你们过这一关！

出于对 SAS 语言的喜爱，我觉得我们应该仔细去读一本好书，一本让我们可以随时拿起来品评的书。不多说了，开卷有益，你绝对不会失望！

——马壮
数学中国网站 (www.madio.net) 站长

专家推荐序一

各位朋友们，大家好！

很高兴诸位决定来学习 SAS。于诸位而言，学习 SAS 或将开辟一个满是机会的新世界，或将有助于您职位的升迁。于我而言，更欣慰的是目睹 SAS 语言正在全球各地蓬勃发展，并将世界顶尖的科技人才关联到一项共同的事业中：通过数据为善。

当我们于 1976 年成立这家公司时完全没有料到，短短 40 余年时间，分析领域竟有如此之进展。当今世界高度互联，我们每个人都以惊人的速度产生大量数据。越来越多的企业见证了基于数据的决策所带来的价值，程序员和数据科学家也成为热需人才，我们因而真正迈进了分析的黄金时代。

在这个新时代，我们的任务是通过揭示洞察来推动进步。诚然我们已经拥有了众多工具来完成使命——原始数据、计算能力、算法等。但我们更需要的是你们这些未来的人才，通晓 SAS，并且可以自信地利用分析应对世间最困难挑战的人才。

此前已经有很多前辈以本书开始了他们的分析之旅。不过诸位更是恰逢其时、更具优势。人工智能、自动化以及互联性的融合力量，正驱动着分析实现各种不可思议的前沿成就。与此同时，我们也彻底革新了 SAS 平台，使其完全开放、灵活，并为未来准备就绪。通过其易用性，SAS 界出现了一个分析专业人士社区，各专业人士可以应用其个人才能为棘手的问题贡献强大的力量。所有这一切都意味着：诸位选择了一个绝佳的时机来学习 SAS。

所以，我热忱地欢迎诸位加入这个全球社区，并诚挚邀请诸位一起共同探索和拥抱未知。诸君及贵国亦都将为此贡献力量。中国乃世界舞台的主角，新技术领域的中流砥柱，同时也是 SAS 重点培育的市场。所以，当诸位在 SAS 社区中寻得一席之地时，请谨记

这个平台将会且一直会持续成长。我深切鼓励诸君融入其中，与 SAS 相伴，共同学习，共同成长！

祝诸君学有所成！

吉姆·古德奈特
SAS 联合创始人兼 CEO
2017 年 8 月，美国

专家推荐序二

近年来人工智能、大数据、物联网以及认知计算等已成为我们业界最热门的研究和发展课题，在这一波风起云涌的分析经济浪潮中，很多数据分析领域的专家、客户或者媒体朋友在与我探讨业务价值和趋势的时候，都会非常关注如何能以更容易入门的方式学习像 SAS 这样的大数据分析软件，以及快速培养更多的分析人才，而这部《The Little SAS Book 中文版》正是应时而生的最佳学习利器。

2017 年，SAS 已经走过 41 年，多年以来我们一直专注在数据分析这个领域，并不断耕耘、持续创新，成为全球分析领域的领导者，同时也成为帮助客户成功并倡导“数据为善”（Data for Good）的行动者。我们资深的专业咨询顾问和同事们帮助用户解决最刁钻复杂的问题，在这一过程中运用了许多复杂的分析模型、算法和解决方案，但这并不意味着 SAS 是一个难以亲近的高科技公司，也不代表 SAS 是一门起点很高的艰深技术。相反的，我们非常开放，也拥抱多元，正如我们近期投资了十亿美元所打造的全新开放平台——SAS[®] Viya™，它不但具有强大的高性能分析功能和基于云平台灵活部署的优势，而且让用户对各种主要开源软件的熟悉变得更容易。

英文版 *The Little SAS Book* 是一本非常经典的 SAS 入门书，多年来在全球 SAS 用户和爱好者圈内广受欢迎，已经更新再版了多次。在 SAS 用户们聚会互动的场合，就曾有多位用户向我表达了对这本书的喜爱，他们充满热忱地表示，正是这本书将他们带入了 SAS 这个美妙的世界。尤其是有些从海外回国的 SAS 粉丝，他们早在 20 年前就开始用 SAS，是最早期的忠实用户，对这本书的中文版特别期待。

众望所归，我也衷心期盼这本书对全球所有的 SAS 爱好者以及大数据分析求知者，尤其是国内的青年才俊们有所启发和帮助。

工欲善其事，必先利其器。祝愿读者们能灵活运用这本实用的技术宝典，掌握开启智慧之门的钥匙，在大数据分析所带来的发展风口上御风而行！

吴辅世
SAS 大中华区总裁
2017 年 8 月，北京



原作者序

1992 年我们开始着手撰写 *The Little SAS Book*, 那时典型的 SAS 程序员都是满书架厚厚的 SAS 手册。就拿《SAS 语言参考》（*SAS Language: Reference*）这本手册来说，厚达 4.5 厘米，重达 2 千克！于是我们决定将我们的新书命名为 *The Little SAS Book*，以示不同。

我们热爱编写 SAS 程序，也希望帮助别人学习编写 SAS 程序。我们都曾在商业界和学术界工作过，深知 SAS 最实用的功能；我们也曾在咨询台工作过，熟知初学者所面临的问题。鉴于此，我们希望我们的书小巧友好、轻松易读、以图述题、并尽可能避免术语。此外，我们确保所有程序完整无误，以便读者在自己的计算机上运行并查看结果。

历经 3 年的努力，*The Little SAS Book* 第 1 版于 1995 年出版。我们乐见其很快成为了 SAS 类图书榜的畅销书，时至今日依然畅销。

自第 1 版问世，SAS 软件改变了许多，*The Little SAS Book* 也随之做了大量的更新。2012 年，我们出版了第 5 版。现如今，这版的中文版将成为本书一个新的里程碑，它将在新的世界获得新的读者。

数据分析需求是全球化的，数据驱动着全球范围内的探索和决策，世界迫切需要优秀的数据分析师来为科学家和决策者提供最高质量的数据。这本书备受赞誉，我们希望您能享受阅读该中文版的乐趣，也希望它能助您达成事业和生活的目标。

祝好！

洛拉 · D. 德尔维奇，苏珊 · J. 斯劳特

The Little SAS Book 作者

2017 年 8 月，美国

译者序

如果一定要给 SASor (SAS 爱好者的自称) 推荐一本圣经级的读物的话，那一定非它莫属：*The Little SAS Book*。

关于原版书

说起这本经典 SAS 书，相信大多数的 SASor 都不会陌生。如果您是首次听说，那容我再唠叨几句。

The Little SAS Book 是洛拉·D. 德尔维奇 (Lora D. Delwiche) 和苏珊·J. 斯劳特 (Susan J. Slaughter) 两位女士的呕心沥血之作。自 1995 年第 1 版问世，至今已逾 20 年，最新版为 2012 年出的第 5 版。20 多年的时间，虽不是沧海桑田、斗转星移，也已是物是人非、今非昔比。然而，两位女士竟然一直坚守不移，深耕细作。从第 1 版到第 5 版，*The Little SAS Book* 一直延续相同的封面设计，变化的是封面颜色、更加精细的章节内容，不变的是作者那份持久的专注和坚持。

如果一件事，做到坚持累积 20 年，迭代精进 20 年，还有何理由不能成为经典？

关于中译本

碰到有意思的人，我们忍不住，说话总要谈及；读到经典的书，我们耐不住，有空总想推荐。对于外文书籍，一个绝妙的推荐方式就是将它翻译出版。动过此念头的 SASor 应该不在少数，我也是其中之一。可惜，起初和 SAS 出版社的沟通，并非那么顺利。如今甚好，《The Little SAS Book 中文版》弥补了这个缺憾。

翻译这本经典，对我们来说如临深渊，如履薄冰。译得好，可算不偏不倚、无所惊奇；译得糟，那便是毁坏经典、遭人鄙夷。SAS 中国也一再叮嘱，此中译本将以 SAS 官方的名义推出。因此，从书名到内容，我们都不得不小心谨慎。

关于书名，我们考虑过尽量本土化的译名，如《SAS 学习手册》《SAS 入门指南》

《SAS 学习精粹》等，也考虑过紧贴原名的译名，如《小小 SAS 书》《迷你 SAS 书》《袖珍 SAS 书》等。但要找一个可以服众的译名与 *The Little SAS Book* 对应，确实很难。既然如此，最终我们索性定为《The Little SAS Book 中文版》，这也是众望所归的结果。至于呼声甚高的《小小 SAS 书》，我们也没舍得浪费。我们用它冠名了我们翻译组的名字：小小 SAS 翻译组。

关于内容，为避免“毁坏经典”，小小 SAS 翻译组在翻译之初便制订了如下的策略：

(1) 每位译者首先自译自校；(2) 每两位译者交叉验证译稿；(3) SAS 官方审核译稿。在经历了两轮交叉验证和两轮审核后，我们终于拿出了一个满意的译本。此外，为了更符合中文读者的习惯，我们将原书中的英文版软件截图全部替换为对应的中文版，代码中的英文注释也做了汉化。所有这些努力，都只是基于一群喜爱本书的 SASor 最基本的要求：经典不能毁于我们之手。当然，我们并非专业的翻译人员，限于时间、精力及水平，本书翻译如有不妥之处，还请诸位读者多多反馈给本书的译者或者审核人员。

本书的译者团队：谷鸿秋（前言～第 1 章以及翻译总体分工规划）、刘欣宇（第 2 章）、孔雁（第 3 章）、李默（第 4 章）、辛岩（第 4~5 章）、李兆钧（第 6 章）、杨志欣（第 7~8 章）、钱明凤（第 9～10 章）、王伟（第 11 章～附录）。此外，杨珂负责了全书的中文版软件截图和代码注释的翻译。

本书的审核团队：马晓丽、王伟、薛剑平、栾思凯、曾秋媚。

致谢

本书的翻译得到了原作者洛拉·D. 德尔维奇 (Lora D. Delwiche) 和苏珊·J. 斯劳特 (Susan J. Slaughter) 的鼓励和支持；原 SAS 大中华区市场总监蒋顺利先生为本书版权引进做了大量的前期协调、沟通和管理工作；清华大学出版社经管畅销书编辑室主任刘洋先生为推动本书的出版，做了大量的努力和付出，在此表示致谢。

本书是小小 SAS 翻译组各位组员集体智慧的结晶，更凝聚了审核团队的心血。因为爱好和热情，我们一起完成了本书，虽然截至本书完稿，我们中的很多人还未曾谋面。就如即将阅读本书的你，我们也未曾谋面，但愿我们“以书会友，友遍天下”。

小小 SAS 翻译组 谷鸿秋

2017 年 8 月 8 日

致 谢

尽管我们竭尽全力来撰写本书，但我们永远不可能独自完成。感谢来自 SAS 公司许多朋友给予的诸多帮助，本书才得以面世。致我们诸位辛勤的审稿人：Amber Elam, Dan Heath, Chris Hemedinger, Anthony House, Sanjay Matange, Lelia McConnell, Sandy Owens, Peter Ruzsa, David Schlotzhauer, Jan Squillace 以及 Grace Whiteis。我们想说：“感谢和我们一起坚持和努力。”致我们的文字编辑 Mary Beth Steinbach 以及设计师 Patrice Cherry 和 Jennifer Dilley，“谢谢你们把本书打造得如此耐看”。致我们的技术出版专家 Candy Farrell，“谢谢整理文中的引号，精写了最后的语句，并且发现缺失的图片”。致我们的市场营销专家 Stacey Hamilton 和 Aimee Rodriguez，“姑娘们，加油！”最后同样要感谢我们的总编辑 Julie Platt，责任编辑 Mary Beth Steinbach 和策划 / 组稿编辑 Stephenie Joyner，你们时时超前、强过 Microsoft Word 并且一蹴而就。

其他很多非 SAS 公司的朋友对本书亦有贡献。在此特别感谢读者，尽管我们看上去有点害羞，但非常乐意在各种会议场合中与诸位相遇。没有你们，我们就没有理由继续坚持写下去。最后，也是最重要的，我们要感谢家人的理解和支持。

SAS 软件介绍

全球数百万人都在使用 SAS 软件——在超过 134 个国家里有 6 万多个软件安装点。SAS（发音为“sass”）既是一家公司，也是一款软件。当人们说起 SAS 时，有时他们是指电脑上运行的 SAS 软件，有时则是指 SAS 公司。

人们经常会问 SAS 到底代表何意？最初，字母 S-A-S 代表 Statistical Analysis System（请勿与 Scandinavian Airlines System, San Antonio Shoemakers 或者 Society for Applied Spectroscopy 混淆）。由于 SAS 产品变得多样化，早年间，SAS 公司便干脆正式放弃了 Statistical Analysis System 的名字，成就了现在简单的缩写称谓：SAS。

SAS 产品 SAS 软件的起源可追溯到 20 世纪 70 年代，它起步于统计分析软件包，但并未止步于此。到 20 世纪 80 年代中期，SAS 已经扩展到图形、在线数据录入以及 C 语言编译器等领域。20 世纪 90 年代，SAS 家族的产品已经发展到包括数据可视化、数据仓库管理以及万维网接口创建等工具。在 21 世纪，SAS 相继设计研发了数据清洗、药物探索与开发、反洗钱等产品，并在不断扩充新的功能。正如 AT&T 现在不仅仅意味着电话和电报一样，SAS 也不仅仅意味着统计。

SAS 系列拥有多样化的产品，而且大多数产品是可集成的。也就是说，它们可以像构件一样共同构建起一个无缝系统。例如，您可以使用 SAS / ACCESS 来读取存储在诸如 Oracle 的外部数据库中的数据；使用 SAS / ETS 软件（业务建模和预测）进行分析；使用 ODS Graphics 生成复杂的图表，然后通过电子邮件把结果发给你的同事；所有的这些都可以通过一个计算机程序完成。要了解有关 SAS 产品的更多信息，请参阅网站 www.sas.com。

操作环境 SAS 软件可广泛运行于各种操作环境。您可以在个人计算机上编写程序，只需更改特定的文件处理语句便可在大型机上运行。由于 SAS 程序是最大限度“可移植”的，因此 SAS 程序员也是“可移植”的。如果您已经了解了一个操作环境中的 SAS，无

须额外学习便可切换到另一个操作环境中。

SASware Ballot SAS 将其营收的很高比例投入研发，SAS 用户每年都可以在 SASware Ballot 上通过投票来帮助确定如何使用这笔资金。该投票是一个新特性和增强功能的建议列表，所有 SAS 用户都有资格进行投票，从而影响 SAS 软件未来的发展。您甚至可以发送电子邮件给 suggest@sas.com，向 SASware Ballot 提出自己的建议。

关于本书

何人需要此书 本书适用于商业、政府和学术等领域的所有 SAS 新手，或者是任何使用 SAS 软件进行数据分析的人员。无论您是否拥有 SAS 使用经验，本书都可帮助您学习未知技术，作为参考书亦有裨益。

本书涵盖内容 本书以大量实例、清晰简明的解释以及尽可能少的术语来介绍 SAS 语言，且大部分的功能均来自 Base SAS。Base SAS 包含了所有程序员所使用的核心功能。但也有例外，例如第 9 章包含了使用 SAS/STAT 模块的过程；此外还有第 2 章及第 10 章中，介绍从其他软件导入导出数据，其中一些方法要求使用 SAS/ACCESS Interface to PC Files。

我们努力介绍初学者可能需要的所有 Base SAS 特性。有些读者可能会对某些主题的出现感到惊讶，例如宏，虽然宏通常被认为是高级技能，有时候新手同样需要用到，因此我们也做了介绍。但这并不是说您需要了解本书的全部内容，您仅需阅读能够解决您问题的部分章节即可。即便是从头到尾通读过此书，当碰到新的编程挑战时，您仍可能需要重启记忆，回来温习相关内容。

本书未涉及的内容 SAS 零基础者亦可阅读本书，但您必须对本地计算机和运行环境有所了解。从一个操作环境转换到另一个操作环境时，SAS 语言几乎是相同的，但有些差异也不可避免。例如，每个操作环境都有不同的文件存储机制和访问方式，一些操作环境比其他操作环境具有更多的交互计算能力，或者您的公司可能有规章制度限制打印文件的大小。本书会尽可能地回答操作环境的相关疑问，但没有任何一本书可以回答与本地系统相关的全部问题，因此您必须对自身的操作环境有一定实践基础，或碰到问题时有可求助之人。

本书并非是 SAS 帮助文档以及众多其他 SAS 出版物的替代品，本书中未能涉及的细节内容，您可以从其他相关资料里进行了解和学习。

我们仅仅讨论了众多 SAS 统计过程中的个别事例。幸运的是，众多的统计过程可以共享语句、选项和输出，因此，介绍的这几个过程可以看作是其他过程的引入。一旦您阅读了第 9 章，自然而然就会熟悉其他统计过程。

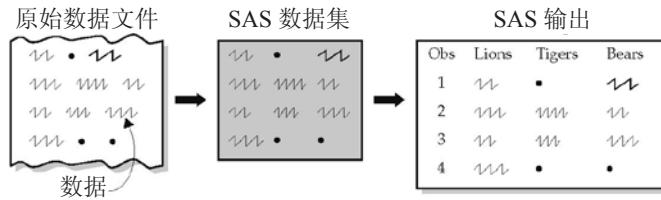
遗憾的是，本书无法对诸如自由度、交叉和嵌套效应等统计概念进行深入介绍。统计检验要有效，就必须满足数据的一些基本假设。实验设计和模型的选择也至关重要。结果的解读往往也很困难，且偏主观。我们假定对统计计算感兴趣的读者对统计学已经有了一定的了解。

希望使用统计过程但不熟悉这些统计概念的读者可以咨询统计学家、寻找统计入门资料，或者更好的办法是学习统计课程。

模块化章节 我们撰写本书的目的就是让学习 SAS 的过程尽可能轻松愉快。当然，我们也需要正视现实：SAS 是一个宏大的话题，您可能曾驻足于满架的 SAS 出版物前挠头苦恼，抑或盯着满屏的文档直到两眼模糊。我们无法将全部 SAS 内容压缩到这本小书中，但我们可以将话题浓缩成短小、易读的章节。

本书每个章节都是一个完整的主题，您可以跳过不适用的部分。当然，我们认为每个章节都很重要，否则就不会在此一一介绍，但您无须通晓全书即可完成相关工作。我们认为把主题以简短、易懂的章节形式呈现，学习 SAS 会更加简单有趣，正如一日三餐要好过一周一次的暴饮暴食。

插图 我们将尽可能用插图来标识章节中的内容或者帮助解释主题。带波纹边框的表示原始数据文件；带平滑边框的表示 SAS 数据集。框内的波纹（ $\backslash\backslash$ ）表示数据（任何原始数据），句点（•）表示缺失值。这些框之间的箭头则表示：该章节解释了如何从一个框里的数据变成另一个框里的数据。有些章节有标明打印输出的插图。这些插图看起来像印有页眉的一堆纸片。



排版约定 SAS 并不区分程序中字母的大小写，您可以随意使用大小写编写程序。本书中，我们区别使用大小写是为了提示一些隐含信息。左边的语句展示的是程序的语法，

或者一般形式；而右边展示的则是可能出现在 SAS 程序中的实际语句。

语法

```
PROC PRINT DATA = data-set-name;  
VAR variable-list;
```

示例

```
PROC PRINT DATA = bigcats;  
VAR Lions Tigers;
```

注意：关键词 PROC PRINT，DATA 和 VAR 在左右两边都相同，但语法中的描述性词语 *data-set-name* 和 *variable-list* 已经被替换为示例中的数据集名和变量名。

在本书中，所有 SAS 关键词全部用大写字母。关键词是 SAS 的指令，且必须拼写正确。任何小写斜体内容都是该语句中当前位置的描述信息，而不是您实际要输入的内容。任何小写或混合大小写的字母（不是斜体字）都是由程序员来编写的，例如变量名称、SAS 数据集的名称、注释和标题。有关 SAS 名称大小写意义的进一步讨论，请参见第 1.2 节。

缩进 本书包含了许多 SAS 程序代码，每段程序代码都是完整可执行的，且程序代码的格式易读易理解。不过 SAS 非常灵活，您无须用这种格式编写您的程序代码，但若注意这些细节，您的程序代码会更易于阅读。当您需要回溯、解读几个月或几年之前的程序时，易读的程序代码可节约您的时间，或者是您以每小时 100 美元聘请的咨询顾问的时间。

缩进首条语句之后的所有其他语句可以体现程序的结构，这个简单的方法可增强程序的可读性，也是我们应该努力养成的好习惯。SAS 并不在意语句的起始位置，即便是所有语句都在一行也无所谓。下面的程序中，缩进了 INFILE 和 INPUT 语句，这表明它们从属于 DATA 语句。

```
*从文件中读取动物的体重。打印结果;  
DATA animal;  
    INFILE 'C:\MyRawData\Zoo.data';  
    INPUT Lions Tigers;  
RUN;  
  
PROC PRINT DATA = animals;  
RUN;
```

本书使用的数据与程序 您可以通过任何一位作者的主页 support.sas.com/delwiche 或者 support.sas.com/slaughter 获取本书示例数据和程序。在主页上，您可以选择示例代码和数据来重现本书中包含的数据和程序。

最后，我们尽力让本书具备可读性，甚至可以轻松阅读。一旦掌握好本册小书的内容，您将不再是 SAS 新手。

最新特性

乍看之下，本书的第五版和第四版太过相似。然而，正如人们常说的那样，表象常常具有欺骗性。事实上，本版的 150 个章节几乎都或多或少进行了修订。

许多章节只是以新的默认格式(HTML)展示输出，但其他章节则是全新撰写。SAS 9.2 推出的 ODS Graphics 产品现在已经成熟，我们觉得有必要用完整的一章来介绍它。自从第二版以来，我们第一次在有关宏编程的那一章中增加了新的章节。当然，对于喜欢以前默认格式(文本)输出的用户，我们也增加了如何将结果输出到 LISTING 上的内容。

我们对本版非常满意，认为它是迄今为止最好的一版，并且希望您也能认同。以下是按章节列出的新主题：

章节	特性
1.6, 1.9	结果查看窗口现在是默认的输出窗口
2.8	\$UPCASE., STIMER. 和 COMMAX. 格式
2.17	PROC IMPORT 的 MIXED=YES 选项
3.4, 3.8, 3.9	YRDIF 函数的 AGE 参数允许计算精确年龄
4.4	PROC SORT 的 SORTSEQ=LINGUISTIC 选项可实现忽略大小写的排序
4.7	\$UPCASE., DTDATETIME., EUROX. 和 ERCENT. 格式
4.10	PROC MEANS 的 MAXDEC= 和 MISSING 选项
4.12	PROC FREQ 的 MISSPRINT 选项
5.4	文本输出的 LISTING 目标
7.3	拼接宏变量和其他文本
8.2	ALPHA=, DATALABEL=, DISCRETEOFFSET=, LIMITSTAT=, MISSING 以及 GROUPDISPLAY= 条形图选项
8.3	BINSTART=, BINWIDTH=, NBINS= 和 TRANSPARENCY= 直方图选项
8.3	TRANSPARENCY= 密度图的选项

章节	特性
8.4	EXTREME, GROUP=, MISSING 以及 TRANSPARENCY= 箱线图选项
8.5	DATALABEL=, NOMISSINGGROUP 以及 TRANSPARENCY= 散点图选项
8.6	CURVELABEL=, DATALABEL=, NOMISSINGGROUP, 以及 TRANSPARENCY= 折线图选项
8.7	ALPHA, CLI, CURVELABEL=, NOLEGCLI 以及 CLMTRANSPARENCY= 拟合曲线选项
8.8	坐标轴的 GRID 选项
8.9	KEYLEGEND 和 INSET 语句
8.10	FILLATTRS=, LABELATTRS=, LINEATTRS=, MARKERATTRS= 以及 VALUEATTRS= 控制图片属性选项
8.11	创建面板图的 PROC SGANEL
8.12	申明图片属性和保存输出图片的选项
9.4, 9.5	均数比较的 PROC TTEST
9.7	PROC FREQ 的 AGREEPLOT, RELRISKPLOT 以及 RISKDIFFPLOT 图



SAS Studio、SAS University Edition 及 SAS OnDemand for Academics

您可以在多个界面下使用 SAS。本书初印之时，SAS Windows 环境（有时称为“显示管理器”）和 SAS Enterprise Guide 是两个主要的界面。如今 SAS Studio 已是 SAS 的一个重要新界面。我们为使用 SAS Studio 的读者撰写了一份本书的网上补充说明以备您之需要。

SAS Studio 已包含在 Base SAS 的许可中，它是 SAS University Edition 的界面，也是 SAS OnDemand for Academics 的默认界面。SAS University Edition 和 SAS OnDemand for Academics 均可免费用于非商业用途。

SAS Studio 是 SAS 的网页版界面。您在 SAS Studio 环境下编写程序、提交到 SAS 服务器后，结果会反馈到 SAS Studio 会话。SAS 服务器可以是本地机，也可以是远程 SAS 服务器。SAS University Edition 采用 SAS Studio 作为 SAS 服务器的接口，SAS 服务器运行于本机安装的虚拟 Linux 服务器。SAS OnDemand for Academics 运行于 SAS 公司的 Linux 服务器上。不管您的计算机是何种操作系统（比如 Windows, OS X for Macs 或者 Linux），SAS University Edition 和 SAS OnDemand for Academics 都会运行于 Linux 系统。

无论您使用何种界面，SAS 编程语言均一样。因此，本书的大部分章节都是准确且时新的，不过，部分章节（1.6 ~ 1.12 节）是展示如何使用 SAS Windows 环境的。如果使用 SAS Studio 作为界面，您大可忽略这些章节转而阅读线上的补充说明材料。

文档 “Using the SAS Studio Interface: Supplement to The Little SAS Book, Fifth Edition” 可从本书任何一位作者的主页下载：<http://support.sas.com/publishing/authors>.



目 录

第 1 章 SAS 软件使用入门	001
1.1 SAS 语言	002
1.2 SAS 数据集	004
1.3 DATA 步和 PROC 步	006
1.4 DATA 步的内置循环	007
1.5 选择提交 SAS 程序的模式	009
1.6 SAS 窗口环境下的窗口和命令	011
1.7 在 SAS 窗口环境中提交程序	013
1.8 阅读 SAS 日志	016
1.9 查看结果	018
1.10 SAS 数据逻辑库	020
1.11 在 VIEWTABLE 窗口中查看数据集	024
1.12 用 SAS 资源管理器查看数据集属性	026
1.13 使用 SAS 系统选项	028
第 2 章 导入数据到 SAS	031
2.1 导入数据到 SAS 的方法	032
2.2 使用 VIEWTABLE 窗口输入数据	034
2.3 使用导入向导读取文件	036

2.4 指定原始数据位置.....	039
2.5 读取空格分隔的原始数据.....	042
2.6 读取按列排列的原始数据.....	044
2.7 读取非标准格式的原始数据.....	046
2.8 常用输入格式.....	049
2.9 混合的输入样式.....	051
2.10 读取杂乱的原始数据.....	053
2.11 为每个观测读取多行原始数据.....	055
2.12 从每行原始数据读取多个观测.....	058
2.13 读取原始数据文件的一部分.....	059
2.14 在 INFILE 语句中使用选项控制输入.....	061
2.15 使用 DATA 步读取分隔文件	064
2.16 使用 IMPORT 过程读取分隔文件.....	066
2.17 使用 IMPORT 过程读取 Excel 文件.....	069
2.18 临时和永久 SAS 数据集	071
2.19 通过 LIBNAME 语句使用永久 SAS 数据集	074
2.20 通过直接引用使用永久 SAS 数据集	076
2.21 列出 SAS 数据集中的内容	079
第 3 章 使用数据	083
3.1 创建和重定义变量.....	084
3.2 使用 SAS 函数	086
3.3 常用 SAS 字符函数	088
3.4 常用 SAS 数值函数	090
3.5 使用 IF-THEN 语句	092
3.6 用 IF-THEN/ELSE 语句分组观测	095
3.7 提取数据的子集.....	097
3.8 使用 SAS 日期	099
3.9 常用日期输入格式、函数和输出格式.....	102

3.10 使用 RETAIN 语句与求和语句	104
3.11 利用数组简化程序	106
3.12 使用变量名列表的快捷方式	108
第 4 章 排序、打印和汇总数据	113
4.1 使用 SAS 过程	114
4.2 使用 WHERE 语句在过程中生成子集	116
4.3 使用 PROC SORT 对数据排序	118
4.4 更改字符数据的排序顺序	120
4.5 使用 PROC PRINT 打印数据	123
4.6 使用输出格式更改打印值的外观	125
4.7 可供选择的标准输出格式	127
4.8 使用 PROC FORMAT 创建自己的输出格式	130
4.9 编写简单的自定义报表	132
4.10 使用 PROC MEANS 汇总数据	134
4.11 将汇总统计量写入 SAS 数据集	136
4.12 使用 PROC FREQ 为数据计数	139
4.13 使用 PROC TABULATE 生成数据报表	141
4.14 将统计量添加到 PROC TABULATE 输出	143
4.15 美化 PROC TABULATE 输出	145
4.16 更改 PROC TABULATE 输出的表标题	147
4.17 为 PROC TABULATE 输出的数据单元格指定多种输出格式	150
4.18 使用 PROC REPORT 生成简单输出	151
4.19 在 PROC REPORT 中使用 DEFINE 语句	153
4.20 使用 PROC REPORT 创建汇总报表	156
4.21 在 PROC REPORT 输出中添加汇总分割	158
4.22 在 PROC REPORT 输出中添加统计量	160
4.23 在 PROC REPORT 输出中添加计算变量	162
4.24 在过程步中使用用户自定义输出格式分组数据	165

第 5 章 使用输出交付系统 (ODS) 增强输出 169

5.1 初识输出交付系统.....	170
5.2 追踪和选择过程步输出结果.....	172
5.3 从过程步输出中创建 SAS 数据集	174
5.4 创建文本输出.....	177
5.5 创建 HTML 输出	179
5.6 创建 RTF 输出	181
5.7 创建 PDF 输出	184
5.8 自定义标题和脚注.....	186
5.9 通过“STYLE=” 选项自定义 PRINT 过程输出	188
5.10 通过“STYLE=” 选项自定义 REPORT 过程输出	190
5.11 通过“STYLE=” 选项自定义 TABULATE 过程输出.....	192
5.12 在输出中添加信号灯效果.....	195
5.13 样式属性列表.....	197

第 6 章 修改和合并数据 201

6.1 使用 SET 语句修改数据集	202
6.2 使用 SET 语句堆叠数据集	204
6.3 使用 SET 语句交错连接数据集	206
6.4 使用一对一匹配合并数据集.....	208
6.5 使用一对多匹配合并数据集.....	211
6.6 合并汇总统计量和原始数据.....	213
6.7 合并总计与原始数据.....	215
6.8 通过事务更新主数据集.....	217
6.9 使用 OUTPUT 语句输出多个数据集.....	219
6.10 使用 OUTPUT 语句将一条观测变为多条观测.....	222
6.11 使用 SAS 数据集选项	224
6.12 使用“IN=” 选项追踪和选择观测	226

6.13 使用“WHERE=”选项选择观测	228
6.14 使用 PROC TRANSPOSE 将观测转置为变量	231
6.15 使用 SAS 自动变量	233
第 7 章 使用 SAS 宏编写灵活的代码	237
7.1 宏概述	238
7.2 用宏变量替换文本	239
7.3 拼接宏变量与其他文本	242
7.4 使用宏创建模块代码	244
7.5 向宏添加参数	246
7.6 编写带条件逻辑的宏	248
7.7 使用 CALL SYMPUT 编写数据驱动程序	251
7.8 调试宏错误	253
第 8 章 可视化数据	257
8.1 ODS 图形概述	258
8.2 绘制条形图	260
8.3 绘制直方图和密度曲线	262
8.4 绘制盒形图	264
8.5 绘制散点图	266
8.6 绘制序列图	268
8.7 绘制拟合曲线	270
8.8 控制坐标轴和参考线	272
8.9 控制图例和插入项	274
8.10 自定义图形属性	276
8.11 绘制面板图形	279
8.12 指定图像属性和保存图形输出	281

第 9 章 基本统计过程的使用方法.....285

9.1 使用 PROC UNIVARIATE 检验数据的分布	286
9.2 使用 PROC UNIVARIATE 创建统计图形	288
9.3 使用 PROC MEANS 生成统计量	290
9.4 使用 PROC TTEST 检验样本均值	292
9.5 使用 PROC TTEST 绘制统计图形	294
9.6 使用 PROC FREQ 检验分类数据	297
9.7 使用 PROC FREQ 创建统计图形	299
9.8 使用 PROC CORR 检验数据的相关性	301
9.9 使用 PROC CORR 创建统计图形	303
9.10 使用 PROC REG 进行简单回归分析	305
9.11 使用 PROC REG 创建统计图形	308
9.12 使用 PROC ANOVA 进行单因素方差分析	310
9.13 理解 PROC ANOVA 的输出	313

第 10 章 导出数据.....317

10.1 数据导出方法.....	318
10.2 使用导出向导生成文件.....	319
10.3 使用 EXPORT 过程导出带分隔符的文件	322
10.4 使用 EXPORT 过程导出 Microsoft Excel 文件	324
10.5 使用 DATA 步导出原始数据文件	326
10.6 使用 ODS 生成带分隔符的文件和 HTML 文件	329

第 11 章 调试 SAS 程序.....333

11.1 编写有效的 SAS 程序	334
11.2 修复无效程序	336
11.3 查找缺失的分号	338

11.4 提示: INPUT 语句到达一行的末尾.....	341
11.5 提示: LOST CARD	343
11.6 提示: 无效的数据.....	345
11.7 提示: 生成缺失值.....	347
11.8 提示: 数值已转换为字符 (或反之)	349
11.9 DATA 步产生错误结果而没有错误消息	352
11.10 错误: 选项无效、选项无法识别、语句无效.....	354
11.11 提示: 变量未初始化 / 错误: 变量未找到.....	357
11.12 SAS 截断字符型变量	359
11.13 SAS 在程序的中间停止	361
11.14 SAS 耗尽内存或磁盘空间	363
附录 从 SQL 到 SAS	367

1

第1章 SAS 软件使用入门

诚实的故事，即使以平淡的方式讲，也传得最快。^①

——威廉·莎士比亚

^① 出自《理查三世》，作者威廉·莎士比亚。公共领域。

1.1 SAS 语言

诸多软件应用不是菜单驱动，就是命令驱动（即输入命令，然后查看结果），但 SAS 两者都不是。在 SAS 中，你用语句来写一系列指令，这些指令被称为 SAS 程序。SAS 程序传达你想要做的事情，而 SAS 程序是由 SAS 语言写成的。SAS 也有一些菜单驱动的前端应用，如 SAS Enterprise Guide，这使 SAS 看起来像是点击式的软件。不过这些前端应用仍然是用 SAS 语言来写程序的。如果你尝试用 SAS 语言自己来编程的话，那会更加灵活。可能你最不想做的事就是去学习一门新的语言，但请相信，虽然 SAS 语言和你熟悉的语言（无论是英语还是 JAVA）有相似之处，但学习 SAS 要容易得多。

SAS 程序 SAS 程序就是一系列按顺序执行的语句。语句向 SAS 发出信息或指令，且必须置于程序中适当的位置。一段 SAS 程序就如同现实生活中我们去一趟银行，你走进银行，然后排队，最后来到柜员窗口前，告知要办理的业务。你所说的话可按程序形式写下来：

我想取现金。

我的账户是 0937。

我要取 200 美元现金。

请给我 5 张 20 美元，2 张 50 美元。

请注意，你首先说明要办理的业务，然后向柜员提供所有所需信息。后续语句的顺序可能并不重要，但你必须首先总体上说明要办理的业务。例如，你不会一上来就对柜员说：“给我 5 张 20 美元，2 张 50 美元。”这不仅是不礼貌的举止，而且很可能会让柜员心里扑通一跳。你必须确保后续语句都围绕第一句展开。当你从活期存款账户取现时，你不会说：“我想要你们这里最大的保险箱。”此话在“我想开设一个保险箱”后面说才合适。SAS 程序就是一系列有序的 SAS 语句，这正如你去银行时使用的一系列有序指令一样。

SAS 语句 同其他任何语言一样，SAS 编程也需要遵循一些语法规则。幸运的是，相比英语，SAS 编程的规则要少很多，简单很多。最重要的规则是：

每一条 SAS 语句都以分号结尾

这看起来非常简单。小孩子可以逐渐改掉忘记在句末加句号的习惯，但 SAS 程序员好像总是忘记在 SAS 语句末尾加分号。即便是最有经验 SAS 程序员，偶尔也会忘记。

如果你谨记此简单规则，你将领先别人几步。

SAS程序的结构布局 关于如何布局SAS程序，实际上没有任何规则。每行放置一条语句，并利用缩进体现程序的不同部分，这样做的确有助于保持程序整洁，但并非必需。

- SAS语句不区分大小写
- 一条语句可以写在多行（只要不拆分单词）
- 多条语句也可以写在一行
- 语句可以从任意列开始

所以你看，SAS是如此的灵活，灵活到可以编写如此杂乱的程序，以至于没有人可以读懂它们，甚至包括你自己。（当然，此做法我们不推荐。）

注释 你可以在程序中插入注释，以便提高程序的可读性，至于注释中写什么并不重要，因为SAS并不检查注释内容。如果愿意，你甚至可以把你最喜欢的饼干食谱放到注释中。不过，注释通常是用来注解程序的，以便别人更容易读懂你的程序，理解你在做什么以及这么做的原因。

关于注释，有两种样式：一种是以星号(*)开头，分号(;)结尾；另一种是以斜杠星号(/*)开头，星号斜杠(*/)结尾。下面的SAS程序展示了两种注释样式的使用方法：

```
*从文件中读取动物的体重;
DATA animals;
INFILE 'C:\MyRawData\Zoo.data';
INPUT Lions Tigers;
PROC PRINT DATA = animals; /*打印结果 */
```

某些操作环境会将第一列中的斜线星号(/*)作为作业结束的标记，因此在使用此类型的注释时不要将其放在第一列。基于此原因，本书选择星号分号(*;)的注释风格。

编程技巧 初学编程语言的人通常会感到沮丧，因为他们第一次写的程序不能正常运行。编写程序应该从小步推进，不要一开始就写一个长而复杂的程序。如果你从小段程序开始，步步为营，并在推进过程中时时检查结果，这将会大大提升你的编程效率。一些没有报错的程序仍有可能是错误的，这就是为什么即便是没有报错，仍然需要检查结果的重要原因。如果确实发生了错误，也不必担心。大多数程序第一次都不能正常运行，原因无他，就因为你是人类。你会忘记分号、拼错单词、手指放错键盘位置，这是常有的事情。一个小的错误往往会导致一大堆的错误。如果你一点一滴地打造你的程序，当出现错误时，也非常容易更正。

1.2 SAS 数据集

SAS 必须先读入数据，然后才能运行分析、撰写报告以及对数据进行任何其他处理。数据必须以一种特殊的形式，也即 SAS 数据集的形式存储，SAS 才能对其进行分析（例外情况请参见 2.1 章节）。SAS 非常灵活，几乎可以读入任何类型的数据，因此把数据读入 SAS 通常会非常简单。一旦数据读入数据集，SAS 会持续追踪其内容和形态。你所做的只需指定所需的数据集的名称和位置，SAS 就会了解其中的内容。

变量和观测 当然，数据是构成任何数据集的基本成分。在传统 SAS 术语中，数据由变量和观测组成。借用关系型数据库的术语，SAS 数据集也可称为表，观测也可称为行，变量则称为列。表 1-1 是一个包含小数据集的矩形表，每行代表一条观测，而 Id、Name、Height 和 Weight 则代表变量。数据点 Charlie 是变量 Name 的一个值，也是第二条观测的一部分。

表 1-1 矩形表

变量（也称为列）

		Id	Name	Height	Weight
观测 (也称为行)	1	53	Susie	42	41
	2	54	Charlie	46	55
	3	55	Calvin	40	35
	4	56	Lucy	46	52
	5	57	Dennis	44	.
	6	58		43	50

数据类型 原始数据有多种不同的形式，不过 SAS 将其进行了简化。在 SAS 中，只有两种数据类型：数值型和字符型。数值字段就是数值，它们可加可减、可有任意的小数位数、可正可负。除了数字外，数值字段还可包含正号 (+)、负号 (-)、小数点 (.) 以及表示科学计数法的 E。数值型数据之外的其余数据均为字符型数据。字符型数据可以包含数字、字母或特殊符号（如 \$ 或者！），最长为 32 767 个字符长度。

如果变量值包含字母或特殊字符，那它必定是字符型变量。但是，如果它只包含数字，那么它可能是数值型也可能是字符型变量，应该根据如何使用该变量做出决定（如果磁盘空间是一个需要考虑的问题，可以选择根据存储容量做出决定，请参见第 11.14 节）。有时，仅包含数字的数据当作字符型变量比当作数值型变量更有意义。例如，邮政编码

由数字组成，但是加减乘除邮政编码没有任何意义，这样的数值存储为字符型更为合理。在前面的数据集中，Name 显然是一个字符型变量，而 Height 和 Weight 是数值型。但是，Id 可以是数值型，也可以是字符型，这取决于你的选择。

缺失数据 有时候，即便你尽了最大的努力，数据仍有可能不完整，有些观测的个别变量的值可能缺失。此时，缺失的字符变量用空白表示，缺失的数值变量用句点表示(.)。在此前的数据集中，第 5 条观测的 Weight 变量缺失，其位置用句点标记。第 6 条观测的 Name 变量缺失，就用留空表示。

SAS 数据集大小 SAS 9.1 以前，SAS 数据集最多可以有 32 767 个变量。从 SAS 9.1 开始，SAS 数据集里的变量多少仅仅受限于你计算机的可用空间。但是，超过 32 767 个变量的 SAS 数据集无法在 SAS 9.1 之前的版本中使用。至于观测的数量，不管你使用哪个版本的 SAS，都只取决于你计算机处理和存储数据的能力。

SAS 数据集与变量命名规则 你需要为数据集和数据集中的变量命名。能表明数据所含内容的名称，特别是变量名称，会很有帮助。尽管诸如 A、B 和 C 之类的变量名看似不错，编程时也易于输入，但当你 6 个月后回头再看程序时，Sex、Height 和 Weigh 之类的变量名或许更为有用。

在命名变量和数据集时，请遵循以下这些简单的规则：

- 名称的长度不能超过 32 个字符。
- 名称必须以字母或下划线 “_” 开始。
- 名称仅可包含字母、数字或者下划线，不可包含%\$!*&#@^①。
- 名称中的字母大小写均可。

最后一条很重要，SAS 不区分大小写，因此你可以用大写、小写或者大小写混合，只要你看着舒服就行，SAS 不关心大小写。数据集名称 heightweight、HEIGHTWEIGHT 和 HeightWeight 在 SAS 看来都是一样的。与此类似，变量名 BirthDate 与 BIRTHDATE 和 birThDaTe 也是一样的。然而，SAS 变量名有一点不同，那就是 SAS 会记住每个变量名第一次出现时的大小写，打印结果时就会采用这种大小写。基于此原因，本书对 SAS 变量名采用混合式大小写，对其他 SAS 名称采用小写。

SAS 数据集中存储的说明信息 除了实际的数据，SAS 数据集还包含了数据集的相关信息，诸如名称、创建日期以及创建数据集所用的 SAS 版本。SAS 还存储了每个变量

^① 如果指定系统选项 VALIDVARNAME=ANY 并采用名称文字形式 ‘变量名’ N，则特殊字符、空格可用于变量名。自 SAS 9.3 开始，非 Windows 环境下运行时，一些特殊字符可以用于 SAS 数据集名。

的信息，包括名称、标签（如果有的话）、类型（数值或者字符）、长度（或者存储大小）以及在数据集中的位置。这些信息有时也被称为数据集的描述部分，这使得 SAS 数据集可以自我说明。

1.3 DATA 步和 PROC 步



SAS 程序由两个基本部分构成：DATA 步和 PROC 步。典型的程序由 DATA 步起始创建 SAS 数据集，而后将数据传递给 PROC 步进行处理。以下是一个简单的示例程序，使用 DATA 步将英里转换为公里，然后用 PROC 步打印结果：

```
DATA step   [ DATA distance;
              Miles = 26.22;
              Kilometers = 1.61 * Miles;
            ]
PROC step  [ PROC PRINT DATA = distance;
             RUN;
```

DATA 步和 PROC 步都由语句组成。一个步中的语句，少则有一条，多则有几百条。大多数的语句只能在一种步中有效，比如只能在 DATA 步中有效，在 PROC 步中无效，反之亦然。新手常犯的错是将语句错误地用于某种步。如果能谨记以下内容，则不太可能犯此错误：DATA 步读取、修改数据；PROC 步分析数据、执行实用功能以及打印报表。

DATA 步以 DATA 语句开始，而 DATA 语句当然以单词 DATA 开头。DATA 关键字后面紧跟着你命名的 SAS 数据集名称。上例中的 DATA 步创建一个名为 distance 的 SAS 数据集。此外，除了从外部原始文件读取数据，DATA 步还可以包含 DO 循环语句、IF-THEN/ELSE 逻辑语句以及各式各样的数值及字符函数。DATA 步还可以按照你期望的几乎任何方式组合数据，包括连接和匹配合并。

而 SAS 过程则以 PROC 语句开头，该语句由关键字 PROC 及其后的过程名（比如 PRINT、SORT 或者 MEANS）组成。大部分 SAS 过程都只有少量的语句。就像遵循食谱一样，你每次基本上都使用相同的语句和要素。SAS 过程可以胜任从简单的排序、打印，到方差分析、3D 图制作等一系列任务。

在下列情形下，SAS 会结束当前的步：当 SAS 碰到新步时（DATA 步或者 PROC 步），

或者碰到 RUN、QUIT、STOP、ABORT 语句时，抑或是在批处理模式下运行到程序结尾处。RUN 语句通知 SAS 去运行本步中此前所有的行，以及在此之前既不属于 DATA 步，也不属于 PROC 步的少许全局语句。上例的程序中，SAS 碰到 PROC 语句就知道 DATA 步结束了。该 PROC 步以 RUN 语句结束，也即程序的结束。

尽管典型的程序以 DATA 步输入或修改数据开始，而后将数据传递给 PROC 步。但这绝非 DATA 步和 PROC 步混合的唯一模式。正如你可以按任何顺序堆叠积木，你也可以按任何顺序排列 DATA 步和 PROC 步。程序甚至可以只包含 DATA 步或只包含 PROC 步。

总结一下，下表汇总了 DATA 步和 PROC 步的基本差异。

DATA 步	PROC 步
● 以 DATA 语句开始	● 以 PROC 语句开始
● 读取、修改数据	● 完成特定分析或者特定功能
● 创建数据集	● 产生结果或报表

查看上表时请谨记，该表将这些差异简单化了。由于 SAS 特别灵活，DATA 步和 PROC 步之间的区别其实没有那么明显。上表并不意味着 PROC 步绝不创建 SAS 数据集（大多数 PROC 步确实如此），或者 DATA 步绝不产生报表（其实它们也可以）。尽管如此，如果你了解 DATA 步和 PROC 步的基本功能，在编写 SAS 程序时会更加轻松。

1.4 DATA 步的内置循环

DATA 步以一种灵活的方式进行数据读取和修改，你可以对数据的处理方式有更多的控制。不过，DATA 步有一个底层结构，即隐含的内置循环。你不必通知 SAS 去执行这个循环，SAS 会自动执行它。

请谨记：

DATA 步逐行执行语句、逐条处理观测

这个基本概念很少有明确说明。因此，很多新手即便是成为老手也没能自行悟出这个道理。

DATA 步逐行执行的理念非常直白简单，也易于理解。这意味着，默认情况下，SAS 首先执行 DATA 步的第 1 行，然后才执行第 2 行，接下来再执行第 3 行，以此类推。这似乎是常识，但是新手经常碰到麻烦，因为他们在创建某变量之前就试图使用该变量。

如果名为 Z 的变量是 X 和 Y 的乘积，那么最好确保创建 X 和 Y 的语句在创建 Z 的语句之前。

SAS 逐行执行语句容易理解，但 SAS 也是逐条观测执行的，这一点就不那么容易理解了。这意味着 SAS 读取第一条观测，然后针对它从头到尾运行 DATA 步（当然是逐行运行）；然后以同样的方式接着处理第二条观测。以这样的方式，SAS 每次只处理一条观测。

我们用慢动作来看 SAS 程序的运行：SAS 从输入数据集读取第一条观测，然后 SAS 执行 DATA 步处理此条观测。如果 SAS 运行到该 DATA 步末尾且未遇到任何严重错误，则将当前的观测写入新的输出数据集。而后返回到该 DATA 步开始处，以同样的方式继续处理下一条观测。直到最后一条观测写入输出数据集，SAS 结束该 DATA 步执行。如果有下一步的话，继续执行下一步。结束慢动作，恢复正常速度。

图 1-1 说明了 DATA 步处理多条观测的流程。

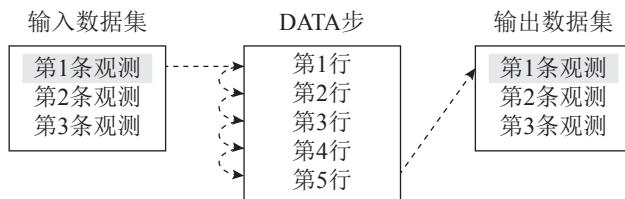


图 1-1

SAS 读取第 1 条观测，然后执行 DATA 步程序的第 1 行，接着执行第 2 行，依此类推，直到 DATA 步末尾。然后，SAS 将此观测写入输出数据集。图 1-1 展示了逐行循环的第一个执行过程。一旦 SAS 处理完第 1 条观测，立即返回到该 DATA 步起始处，继续读取并处理第 2 条观测。当 SAS 读取完最后一条观测时，就会自动停止循环。

打一个比方，DATA 步的处理过程有点像投票。到达投票站时，你排在其他先到的投票人之后。当你到达队伍的最前面时，你要回答标准问题“你叫什么名字？你住在哪里？”接下来，你签下姓名，投出选票。在这个类比中，选民就如同观测，投票过程就好比 DATA 步。选民逐个投票，就好比观测被逐条处理。每个选民的选择都是秘密，偷窥旁人的选票绝对是不受欢迎的。此外，每个选民都以相同的顺序（逐行）完成过程的每个步骤。在你提供姓名和地址之前，你不能投票，一切都必须按正确的顺序完成。

如果这有点过于结构化，SAS 也提供了一些打破逐行执行程序，逐条处理观测的方法，这些方法包括 RETIAN 语句（第 3.10 节）和 OUTPUT 语句（第 6.9 和 6.10 节）。

1.5 选择提交 SAS 程序的模式

目前我们已经讨论了如何编写 SAS 程序，但是仅仅编写程序并不会得到任何结果。就如同写信给国会代表，不邮寄出去也不会有结果。SAS 程序在你提交运行之前也是毫无动静。你可以采取多种方式执行 SAS 程序，但并非每个方法都适用于所有操作环境。请查阅与你的操作环境相关的 SAS 帮助文档，找到适用的方法。你选择运行 SAS 程序的方法将取决于你的偏好以及你的应用程序和环境。如果你在具有大量用户的场所使用 SAS，请多方询问，找出最易接受的运行 SAS 的方法。如果你在自己的个人电脑上使用 SAS，请选择适合你的方法。

SAS 窗口环境 如果在系统提示符下输入 SAS，或单击 SAS 图标，则很有可能进入 SAS 窗口环境（也称为显示管理器）。在这种交互式环境中，你可以编写 SAS 程序，提交程序进行处理，并查看和打印结果。此外，还有许多 SAS 窗口用于执行不同的任务，例如管理 SAS 文件、自定义界面、访问 SAS 帮助文档以及导入导出数据。窗口环境的确切效果取决于所使用的计算机类型、该计算机的操作环境以及启动 SAS 时生效的选项。如果你使用的是个人计算机，则 SAS 窗口环境将与计算机上的其他程序类似，并且许多功能你也很熟悉。图 1-2 显示了微软 Windows 操作系统下的 SAS 窗口环境。

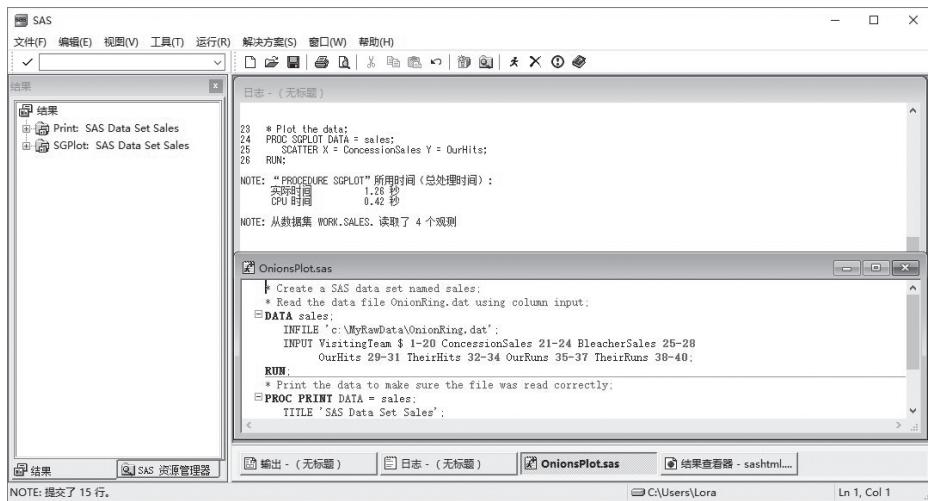


图 1-2

SAS Enterprise Guide 如果你有 SAS Enterprise Guide 软件（仅有 Windows 版），

你可以选择在 SAS Enterprise Guide 里提交你的程序。为此，打开一个“程序”窗口，输入 SAS 程序或打开现有的 SAS 程序。SAS Enterprise Guide（从版本 4.3 开始）的程序编辑器在你键入程序时会自动显示语法帮助，并且有一个程序分析器将生成程序图，以帮助你可视化各部分及其组合架构。你可以选择在本机或在已安装了 SAS 的远程服务器上运行代码。要在远程服务器上运行 SAS 程序，可能还需要安装其他 SAS 软件。此外，SAS Enterprise Guide 可以通过其众多的菜单系统为你编写 SAS 代码。SAS Enterprise Guide 是基于项目的，所以你的所有程序、结果及相应的数据引用都存储在一个项目文件中。图 1-3 显示了 SAS Enterprise Guide 4.3 中的一个项目。

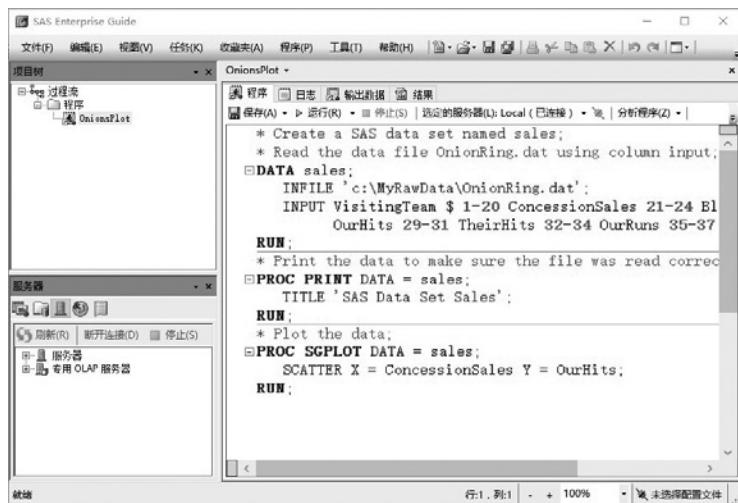


图 1-3

批处理或者后台模式 在批处理或者后台模式下，SAS 程序就是一个文件。你提交这个文件供 SAS 处理，你的 SAS 程序可能立即被执行，也可能会在其他作业后排队。批处理在大型机上使用很普遍，当你的作业在处理时，你仍然可以继续使用你的电脑。更棒的是，你甚至可以去参加棒球比赛，让计算机在你缺席的情况下工作。批处理通常比其他方法成本低，特别适用于大型作业，这些作业可以在非高峰时段进行，这样费用就可以最低。当你的作业完成后，结果将保存于一个或多个文件中，随时可以显示或者打印。

想知道如何以批处理模式提交 SAS 程序，请查阅适用于你的操作环境的 SAS 帮助文档，或者咨询你所在软件安装点的其他 SAS 用户。即便是操作环境相同的软件安装点，

以批处理模式提交作业的方式也可能会有所不同。

1.6 SAS 窗口环境下的窗口和命令

SAS 窗口环境（也称“显示管理器”）采用操作环境的外观，这对你有益，因为这样 SAS 窗口环境的许多方面你都会比较熟悉。但是，由于你可以通过多种方式自定义 SAS 环境，这就让介绍 SAS 窗口环境变得困难，因为我们无法确切地指出你的 SAS 会话的界面和行为。不过，各种操作环境之间都有通用的元素，你可能已经了解那些不同的元素。

SAS 窗口

SAS 窗口有 5 种：结果窗口、资源管理器窗口以及 3 种程序窗口（编辑器、日志、输出）。在 Windows 操作环境中，还有第 6 个窗口，如果你运行的程序产生可打印结果的话，就会出现“结果查看器”。有时窗口不会立即显示，例如，在 Windows 操作环境中，“输出”窗口最初显示在“编辑器”和“日志”窗口后面。还有许多其他 SAS 窗口可用于诸如获取帮助、更改 SAS 系统选项以及自定义 SAS 会话等任务。图 1-4 显示了微软 Windows SAS 会话的窗口，指针在主 SAS 窗口。

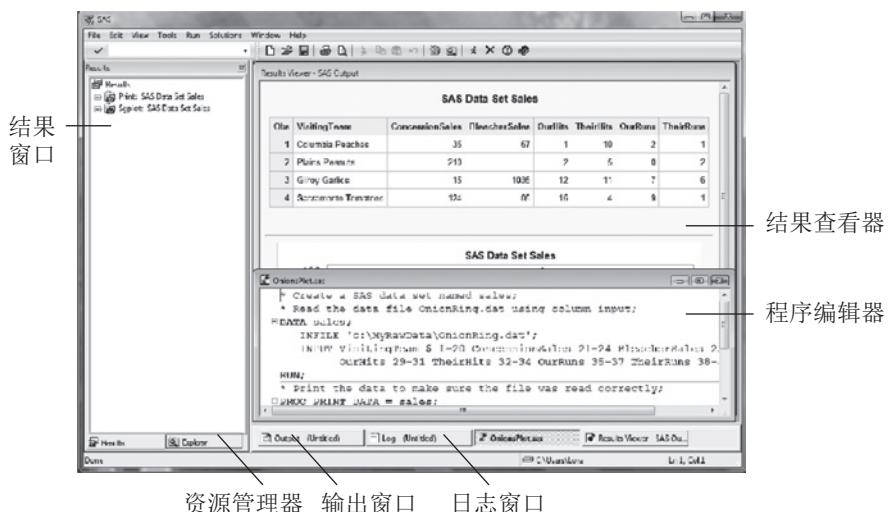


图 1-4

编辑器 这是一种文本编辑器，你可以输入、编辑、提交 SAS 程序，也可以编辑其他文本文件，比如原始数据文件。在 Windows 操作环境中，默认的编辑器是增强型编辑器。增强型编辑器对语法敏感，可以给代码着色，使程序易读，易发现错误。增强型编辑器还可以折叠和扩展程序中的各种步。在其他操作环境下，默认的编辑器是程序编辑器，其特性因 SAS 版本和操作环境而异。

日志 日志窗口包含有关 SAS 会话的提示信息，在提交 SAS 程序后，与程序相关的任何提示、错误或警告以及程序语句本身都将显示在“日志”窗口中。

输出 在 z / OS 操作环境中，所有表格结果将显示在“输出”窗口中。在 Windows 和 UNIX 环境中，“输出”窗口默认不显示任何内容。但是如果你开启了 LISTING 目标（见第 5.4 节），那么结果将显示在“输出”窗口中。

结果查看器 在 Windows 操作环境中，如果程序生成可打印的结果，则会打开“结果查看器”窗口并显示结果。

结果 “结果”窗口就如同“输出”窗口和“结果查看器”窗口的目录。结果树以纲要形式列出每部分结果。

资源管理器 “资源管理器”窗口可以让你轻松访问 SAS 文件和逻辑库。

SAS 命令

SAS 命令可完成多种任务，其中一些可能比较常见，例如打开和保存文件、剪切和粘贴文本以及访问帮助。其他一些命令是 SAS 特有的，比如提交 SAS 程序。你有多达三种方式提交命令：菜单、工具栏以及 SAS 命令栏（或者命令行），图 1-5 显示了在 Windows 操作环境中这三种环境提交 SAS 命令的位置。



图 1-5

下拉菜单 大多数操作环境将在每个窗口顶部或屏幕顶部设有下拉菜单。如果你的菜单位于屏幕顶部，则当你激活不同的窗口（通常通过单击它们）时，菜单将会更改。对于每个窗口，当你用鼠标右击或按中央按钮时就会出现上下文相关的弹出式菜单。

工具栏 工具栏（如果有的话）提供某些命令的快捷方式，这些命令都能在下拉菜

单中找到，但并非所有操作环境都有工具栏。

SAS命令栏 命令栏是可以键入SAS命令的地方。在某些操作环境中，命令栏和工具栏（如上图所示）挨着；在其他操作环境中，你可以使用每个SAS窗口的命令行（通常由“Command =>”标识）。你可以在命令栏中键入的大多数命令也可以通过下拉菜单或工具栏访问。

控制你的窗口 窗口下拉菜单可让你选择屏幕上窗口的放置位置。你还可以通过从窗口下拉菜单中选择窗口或通过单击窗口来激活任何编程窗口。

1.7 在SAS窗口环境中提交程序

费心费力写好SAS程序之后，你自然希望看到结果。如前所述，提交SAS程序的方法有多种。如果使用SAS窗口环境，则可以在窗口环境中完成从编辑程序、提交程序直到查看结果的所有任务。

将程序置入编辑器 你需要做的第一件事就是将程序置入“编辑器”窗口中。你可以在编辑器中键入程序，也可以将程序从文件中放入编辑器窗口。对于编辑器的“编辑”以及“打开文件”命令，你应该不陌生。SAS尽量遵循你的操作环境的惯例。例如，要在编辑器中打开文件，你可以从菜单栏中选择**文件**▶**打开**。对于某些操作环境，工具栏上可能有一个“打开”图标，你也可以选择将剪贴板中的文件粘贴到编辑器中。

提交SAS程序 一旦你的SAS程序出现在编辑器中，你就可以用SUBMIT命令执行它（可执行整个程序或者突出显示的部分）。因操作环境的不同，执行SUBMIT命令的方式会有所不同。首先，单击激活“编辑器”窗口，接下来你可以选择以下操作中的一项执行

单击工具栏上的“提交”按钮 。

在SAS会话的命令行区域输入“SUBMIT”命令，如图1-6所示。

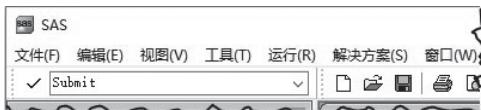


图 1-6

在菜单栏选择**运行**▶**提交**，如图1-7所示。



图 1-7

查看 SAS 日志和结果 在 Windows 操作环境中，提交程序后，该程序将保留在“增强型编辑器”窗口中，程序的结果进入“日志”和“结果查看器”窗口。在 UNIX 环境中，你的结果进入“日志”窗口和单独的 Web 浏览器窗口。而在 z / OS 中，你的结果将进入日志“日志”和“输出”窗口。对于 UNIX 和 z / OS，程序提交后，会从“程序编辑器”窗口中消失。一开始时，看到程序从你的眼前消失，你可能会感到很惊讶。别担心，你花了这么长时间写的程序并没有就此消失。如果程序有任何输出，“结果”窗口就会有新条目。“结果”窗口就如同是“输出”窗口的目录，更多详细信息将在第 1.9 节中介绍。图 1-8 展示了在 Windows 环境中，从增强型编辑器提交程序后屏幕的样子。



图 1-8

你可能无法同时看到所有的窗口。在有些操作环境下，一些窗口会在其他一些窗口前面。在图 1-8 中，“资源管理器”窗口在“结果”窗口后面，“输出”窗口和“日志”窗口在“增强型编辑器”窗口和“结果查看器”窗口后面，通过单击窗口或者其选项卡，或者在命令行区域键入其名称，或者在窗口菜单中选择，均可将其置于最前面。

找回程序 不幸的是，对于大多数人来说，我们的程序并不能每次都完美运行。如果你的程序出现错误，你很可能想要编辑程序并再次运行。如果你使用的是增强型编辑器，那么你的程序将在你提交之后保留在窗口中。但是，如果使用的是“程序编辑器”窗口，则需要使用 RECALL 命令将程序恢复到“程序编辑器”窗口中。你可以通过两种方式执行 RECALL 命令。

1. 确保“程序编辑器”窗口是活动窗口，然后在 SAS 会话的命令行区域敲入 RECALL 命令，如图 1-9 所示。

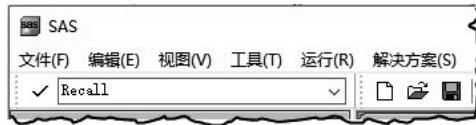


图 1-9

2. 确保“程序编辑器”窗口是活动窗口，然后从菜单栏选择运行 ▶ 重新调用上一次提交，如图 1-10 所示。

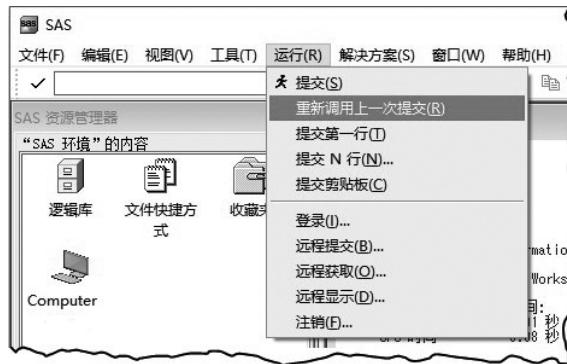


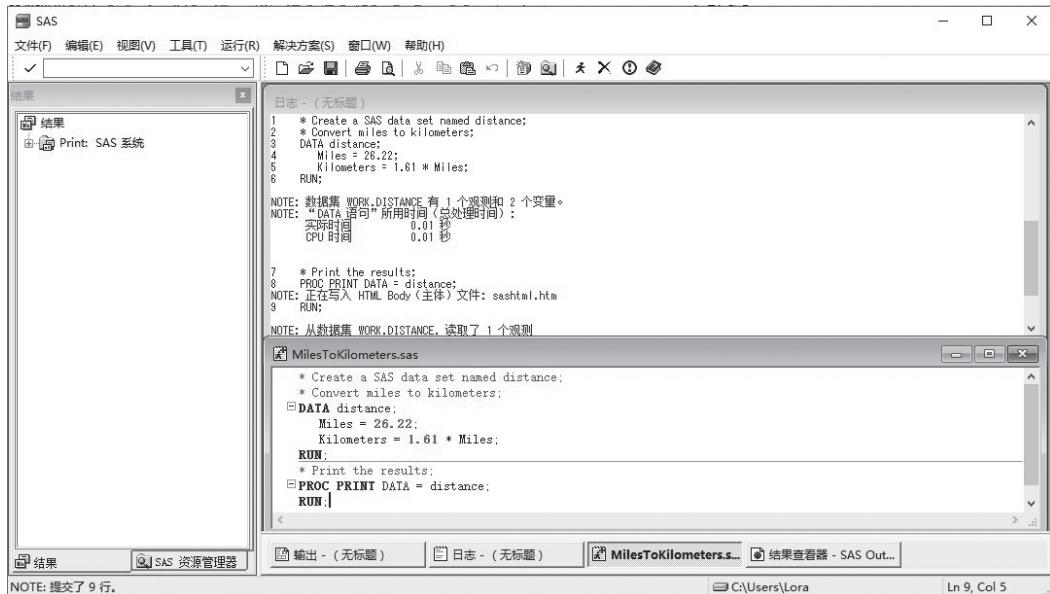
图 1-10

RECALL 命令将返回你提交的最后一个语句块。如果再次使用 RECALL 命令，它将插入倒数第二个提交的语句块，依此类推，直到它找回所有你提交的语句。

1.8 阅读 SAS 日志

每次提交 SAS 程序时，SAS 都会在日志中写入消息。许多 SAS 程序员忽略 SAS 日志直接查看输出结果。这样做可以理解，但这种做法很危险。输出结果有时看似正确，实际上却不正确，这是有可能出现的情况，而且迟早我们都会碰到。及早知道它们是坏结果的唯一方法是检查 SAS 日志。程序正常运行并不意味着结果就是正确的。

SAS 日志的位置 SAS 日志的位置因操作环境、模式（SAS 窗口环境或者批处理）以及本机设置的不同而异。如果在窗口环境中提交程序，SAS 日志默认会显示在“日志”窗口中，正如图 1-11 所示。



The screenshot shows the SAS software interface. At the top is a menu bar with File, Edit, View, Tools, Run, Solution, Window, and Help. Below the menu is a toolbar with various icons. The main area has several windows open:

- 结果 - (无标题)**: This window contains the log output. It shows the SAS code being run, including PROC PRINT statements, and notes about the execution environment.
- MilesToKilometers.sas**: This is the code editor window showing the SAS program code.
- 输出 - (无标题)**, **日志 - (无标题)**, **MilesToKilometers.sas**, **结果查看器 - SAS Out...**: These are tabs at the bottom of the interface.

At the bottom left, it says "NOTE: 提交了 9 行。" At the bottom right, it shows the path "C:\Users\Lora" and "Ln 9, Col 5".

图 1-11

如果在批处理模式下提交 SAS 程序，日志将写到一个文件中，这个文件你可以通过操作环境命令查看和打印。日志文件的名字基本上就是将原程序名字置换一下。例如，如果程序的名字是 Marathon.sas，那日志的名字一定是 Marathon.log。

日志包含的内容 人们倾向于将 SAS 日志视为程序的改写或者只是一堆含混的信息。好吧，我们承认，SAS 日志中确有一些琐碎的技术细节，但也有很多重要的信息。以下是将英里转换为公里并打印结果的简单程序：

```
* 创建一个名字为distance的SAS数据集;  
* 把英里转换为千米;  
DATA distance;  
    Miles = 26.22;  
    Kilometers = 1.61 * Miles;  
RUN;  
* 打印结果;  
PROC PRINT DATA = distance;  
RUN;
```

如果你运行该程序，SAS 将产生如下的类似日志。

```
① NOTE: Copyright (c) 2002-2012 by SAS Institute Inc., Cary, NC, USA.  
NOTE: SAS (r) Proprietary Software 9.3 (TS1M0)  
      Licensed to XYZ Inc, Site 0099999001.  
NOTE: 该会话正在平台 W32_VSPRO 上执行。  
NOTE: "SAS初始化"所用时间:  
      实际时间          1.40 秒  
      CPU时间          0.96 秒  
② 1   * 创建一个名字为distance的SAS数据集;  
2   * 把英里转换成千米;  
3   DATA distance;  
4       Miles = 26.22;  
5       Kilometers = 1.61 * Miles;  
6   RUN;  
③ NOTE: 数据集 WORK.DISTANCE 有1个观测和2个变量。  
④ NOTE: "DATA 语句"所用时间(总处理时间):  
      实际时间          0.03 秒  
      CPU时间          0.03 秒  
② 7   * 打印结果;  
8   PROC PRINT DATA = distance;  
9   RUN;  
NOTE: 从数据集 WORK.DISTANCE. 读取了 1 个观测  
④ NOTE: "PROCEDURE PRINT"所用时间(总处理时间):  
      实际时间          0.01 秒  
      CPU时间          0.00 秒
```

上面的 SAS 日志是 SAS 执行该程序的逐条记录。

- ① 以 SAS 版本和 SAS 软件安装点编号开始。
- ② 包含了原始程序语句，并在左侧添加了行号。
- ③ DATA 步紧跟着注释，注释里包括创建的 SAS 数据集名称（WORK.DISTANCE）、观测数（1）和变量数（2）。匆匆一瞥就可以确保你没有丢失观测，或者是否不小心创建了大量不需要的变量。

④ DATA 步和 PROC 步都会产生关于所用计算机资源的注释。起初你可能丝毫不会在意。但是，如果你是在多用户系统上运行或者拥有大量数据集的长作业时，这些统计信息可能就会引起你的兴趣。如果你想知道为什么你的作业需要运行这么长的时间，那么看一眼 SAS 日志，你就会知道哪些步骤是罪魁祸首。

如有错误消息，它们会在日志中显示，说明 SAS 在何处出错以及执行的操作。你还可能会发现警告和其他类型的提示信息，这些提示信息有时会指示有错误，有时只是提供有用的信息。第 11 章讨论了 SAS 用户碰到的几种常见错误。

1.9 查看结果

如何查看输出取决于所用的操作环境以及提交程序的方式。

SAS 窗口环境 如果在微软 Windows 系统下以 SAS 窗口环境提交程序，那么输出默认会显示在“结果查看器”窗口中，且以 HTML 形式呈现。如果是在 UNIX 环境下，输出仍然默认以 HTML 形式呈现，但是会在一个单独的 Web 窗口呈现。如果是在 z/OS 环境下，输出结果会以文本形式呈现在输出窗口中。

批处理模式 如果以批处理模式提交程序，输出将会保存在计算机上的一个文件中，你可以利用操作环境命令查看这个输出文件（也称为列表）。例如，如果在 UNIX 系统中以批处理方式执行 SAS 程序，那么输出将保存在扩展名为 .lst 的文件中。要查看该文件，你需要使用 cat 或者 more 命令。

结果查看器窗口 当你在微软 Windows 系统下以 SAS 窗口环境提交程序后，程序结果将显示在“结果查看器”窗口中。“结果查看器”窗口自动打开并出现在所有打开的程序窗口的上面。图 1-12 展示了在提交包含 ANOVA（方差分析）过程的程序后“结果查看器”的可能的样子。请注意，“结果查看器”窗口自动滚动到最底部，所以你看到的是该过程输出的结尾。

结果窗口 当你有大量的输出时，“结果”窗口就非常有帮助。“结果”窗口就如同输出内容的目录，它列出了产生输出的每个过程。如果你打开或者展开“结果”树中的过程，就会看到该过程的每一部分输出。要展开结果树，点击加号（+），或者右击结果，然后选择全部展开。

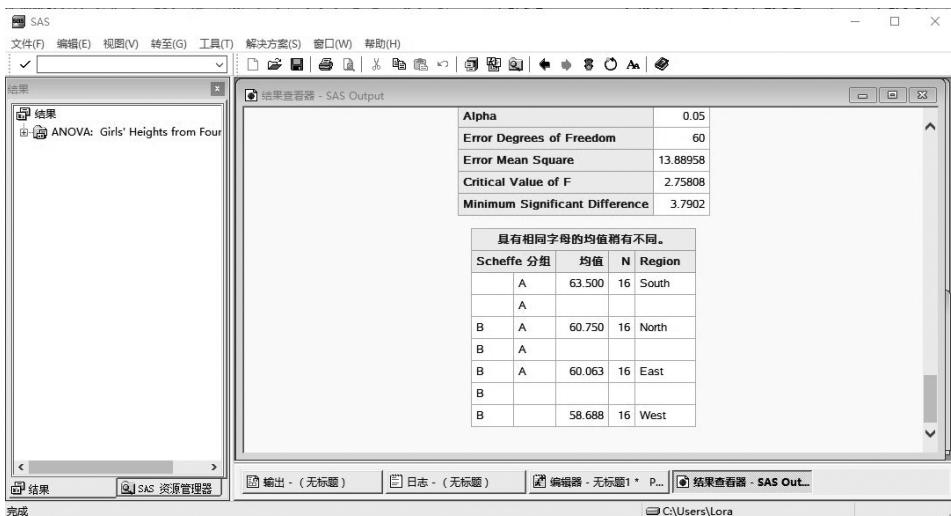


图 1-12

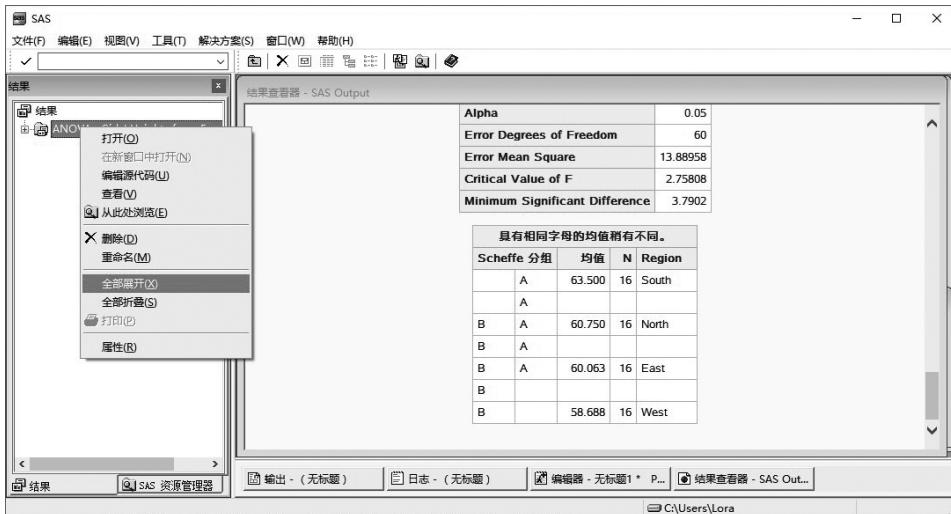


图 1-13

双击你想要查看的输出，它就会出现在“结果查看器”的顶部。图 1-14 展示了双击“结果查看器”窗口的**总体 ANOVA** 条目后“结果查看器”窗口的外观。



图 1-14

1.10 SAS 数据逻辑库

在使用 SAS 数据集前，你需要告诉 SAS 它放在什么地方。你可以通过设置 SAS 逻辑库实现这一点。SAS 逻辑库只是一个存储 SAS 数据集（以及其他类型 SAS 文件）的位置。根据操作环境的不同，SAS 逻辑库可能是计算机上的一个文件夹或者目录，也可能是一个物理位置，如硬盘、闪存盘或者 CD。要设置 SAS 逻辑库，只需给你的逻辑库取名，并将其位置告诉 SAS 即可。设置 SAS 逻辑库有若干种方法，包括使用 LIBNAME 语句（在第 2.18 节至第 2.19 节中介绍）和在 SAS 窗口环境中使用“新建逻辑库”窗口。

启动 SAS 窗口环境时，你会看到基本的 SAS 窗口，如图 1-15 所示，包括“资源管理器”窗口（如果“资源管理器”窗口在“结果”窗口下面，单击其选项卡将其前置）。如果你双击逻辑库图标，资源管理器会打开“当前逻辑库”窗口，并显示所有已定义的逻辑库。要返回到资源管理器的之前窗口，在菜单栏选择视图 > 向上一级，或者单击激活“资源管理器”窗口，然后单击工具栏上的“向上一级”按钮。



图 1-15

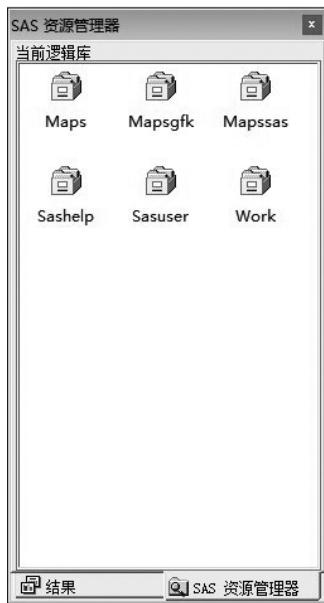


图 1-16

当前逻辑库窗口 打开“当前逻辑库”窗口，你将至少看到三个逻辑库：SASHHELP、SASUSER 和 WORK。可能还有其他特定 SAS 产品的逻辑库（如 SAS/GRAFH 软件的 MAPS 逻辑库），或由你或同事设置的逻辑库，如图 1-16 所示。SASHHELP 逻辑库包含

控制你 SAS 会话的信息以及 SAS 示例数据集。WORK 逻辑库是 SAS 数据集的临时存储位置，它也是默认逻辑库。若你创建 SAS 数据集时不指定逻辑库，那么 SAS 会将其放入 WORK 逻辑库中，然后在结束会话时将其删除。如果你更改 SAS 窗口环境的默认设置，则该信息将存储在 SASUSER 逻辑库中。你还可以将 SAS 数据集、SAS 程序和其他 SAS 文件存储在 SASUSER 逻辑库中。但是，很多人喜欢为他们的 SAS 文件创建一个新的逻辑库。

创建新逻辑库 你可以使用“新建逻辑库”窗口创建新的 SAS 逻辑库。要打开该窗口，从菜单栏选择 **工具** ▶ **新建逻辑库** 或者右击“当前逻辑库”窗口，然后从弹出菜单中选择 **新建**，如图 1-17 所示。



图 1-17

如图 1-18 所示，在“新建逻辑库”窗口中，键入你要创建的逻辑库的名称。该名称被称为逻辑库引用名（libref，即 library reference 的缩写）。逻辑库引用名不能超过 8 个字符，以字母或者下划线开头，且只能包含字母、数字或下划线。在此窗口中，键入名称 BIKES 作为逻辑库引用名。在“路径”字段中，输入要存储数据集的文件夹或目录的完整路径，或单击**浏览 (R) ...** 按钮导航到该位置。如果你不想每次启动 SAS 时都定义

逻辑库引用，则选中“启动时启用”复选框。单击“确定”按钮，然后你的新逻辑库引用将出现在“当前逻辑库”窗口中。



图 1-18

如图 1-19 所示，这是显示了新建的 BIKES 逻辑库的“当前逻辑库”窗口。



图 1-19

1.11 在 VIEWTABLE 窗口中查看数据集

除了列出当前逻辑库和创建新逻辑库，还可以使用 SAS 资源管理器打开 SAS 数据集，以便在 VIEWTABLE 中查看。当编写程序时，及时检查你创建的数据集的正确性，总是有一个不错的主意。VIEWTABLE 是查看 SAS 数据集的一种方式。

双击上一节中显示的“资源管理器”窗口中的逻辑库图标，这样会打开“当前逻辑库”窗口，显示系统中当前定义的所有逻辑库。如果双击逻辑库图标，SAS 会打开一个“内容”窗口，显示特定逻辑库中的所有 SAS 文件，如图 1-20 所示。

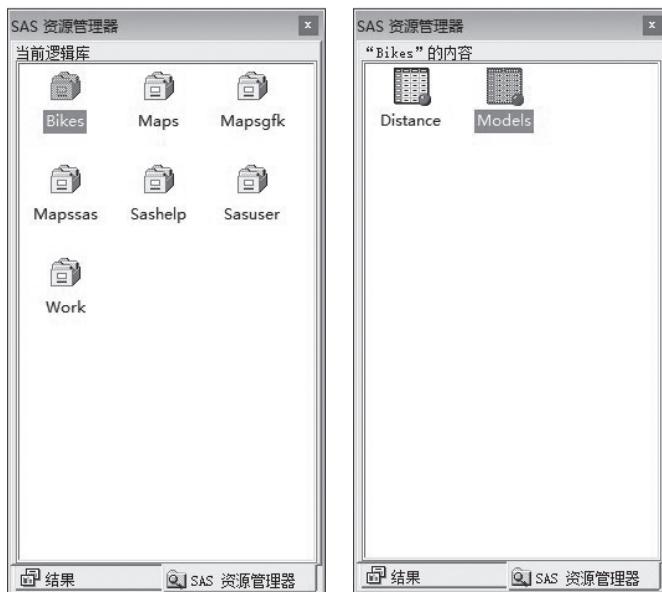


图 1-20

要返回资源管理器中之前的窗口，从菜单栏中选择 **视图 > 向上一级**，或者单击激活“资源管理器”窗口，然后点击工具栏的“向上一级”按钮 。

内容窗口 该窗口显示逻辑库的内容。SAS 数据集由一个小数据表和红色球组成的图标表示。图 1-20 右侧显示的逻辑库库中包含两个名为 MODELS 和 DISTANCE 的数据集。如果双击某个数据集，SAS 将打开一个 VIEWTABLE 窗口显示该数据集。（如果你还没有自己的 SAS 数据集，你可以查看 SASHELP 逻辑库中 SAS 提供的示例数据集，SASHHELP 库中的 CLASS 数据集是查看数据集的一个不错的选择。）

更改列标题 默认情况下，VIEWTABLE 使用变量标签作为列标题，或者如果变量没有标签，则显示变量名称。有时你可能希望看到实际的变量名称而不是标签，要实现这一点，请单击激活 VIEWTABLE 窗口，然后从菜单栏中选择视图 ▶ 列名。图 1-21 是显示列（也称为变量）名称而非标签的 MODELS SAS 数据集。

	Model Name	Type of Bicycle	List Price	Frame Composition
1	Black Bora	Track	\$796.00	Aluminum
2	Delta Breze	Road	\$399.00	CroMoly
3	Jet Stream	Track	\$1,130.00	CroMoly
4	Mistral	Road	\$1,995.00	Carbon Comp
5	Nor'easter	Mountain	\$899.00	Aluminum
6	Santa Ana	Mountain	\$459.00	Aluminum
7	Scirocco	Mountain	\$2,256.00	Titanium
8	Trade Wind	Road	\$759.00	Aluminum

图 1-21

更改列标题 默认情况下，VIEWTABLE 使用变量标签作为列标题，或者如果变量没有标签，则显示变量名称。有时你可能希望看到实际的变量名称而不是标签，要实现这一点，请单击激活 VIEWTABLE 窗口，然后从菜单栏中选择查看 ▶ 列名称。图 1-21 显示列（也称为变量）名称而非标签的 MODELS SAS 数据集。

列选项 如果右击列标题，则弹出菜单中将显示几个选项，如图 1-22 所示。你可以控制颜色、字体和查看列属性。你可以选择按列中的值对数据进行排序。在非编辑模式下，可以选择创建包含已排序数据的新数据集。你也可以隐藏或冻结列。如果你选择隐藏列，则数据将不会在当前的 VIEWTABLE 会话中显示。要取消隐藏列，请从菜单栏中选择数据 ▶ 隐藏/取消隐藏以打开“隐藏/取消隐藏”窗口。在此窗口中，你可以更改所有列的可见性。当你选择冻结某列时，即使你向右滚动滚动条，该列及其左侧的每一列都始终可见。

Model	Colors...	Price	Frame
1 Black Bora		\$796.00	Aluminum
2 Delta Breze		\$399.00	CroMoly
3 Jet Stream		\$1,130.00	CroMoly
4 Mistral		\$1,995.00	Carbon Comp
5 Nor'easter		\$899.00	Aluminum
6 Santa Ana		\$459.00	Aluminum
7 Scirocco		\$2,256.00	Titanium
8 Trade Wind		\$759.00	Aluminum

图 1-22

1.12 用 SAS 资源管理器查看数据集属性

SAS 数据集的“属性”窗口包含了一些非常有用的信息，例如数据集的创建日期和时间、观测个数、所有变量名以及变量属性。“属性”窗口包含了与第 2.21 节中 CONTENTS 过程产生输出类似的信息。

打开“属性”窗口 要打开“属性”窗口，请先从“资源管理器”窗口中双击逻辑库图标，然后双击包含 SAS 数据集的逻辑库。SAS 将在“资源管理器”窗口中显示逻辑库的内容。如图 1-23 所示，右击数据集的图标，然后从弹出菜单中选择“属性”，此时将打开“属性”窗口，其中的“常规”选项卡位于最前面。图 1-24 显示了微软 Windows 操作环境中“属性”窗口的外观。



图 1-23

“常规”选项卡 此窗口显示有关数据集的信息，例如其创建日期以及行数（或观测数）和列数（或变量数）。

“列”选项卡 如果单击“列”选项卡，则 SAS 将显示有关该数据集中的列（或变量）的信息，如图 1-25 所示。变量名称、类型和长度以及分配给变量的输入输出格式都将显示。变量标签也显示在此窗口中，但要查看它们，你需要向右拖动滚动条。



图 1-24

The screenshot shows the 'Properties' dialog box for the 'Models' table, specifically the 'Columns' tab. The title bar reads 'Bikes.Models' 属性. The tabs at the top are 常规 (General), 详细信息 (Details), 列 (Columns), 索引 (Indexes), 完整性 (Integrity), and 密码 (Password). The Columns tab is selected. A search bar labeled '查找列名:' (Find Column Name:) is present. Below is a table listing the columns:

列名	类型	长度	输出格式	输入格式	标签
Model	文本	12	\$12.	\$12.	Model Name
Type	文本	8	\$8.	\$8.	Type of Bike
Price	数字	8	DOLLAR12.2	F8.	List Price
Frame	文本	13	\$13.	\$13.	Frame Color

At the bottom are buttons for 确定 (OK), 取消 (Cancel), and 帮助 (Help).

图 1-25

如果你的数据集中有很多变量，使用排序和查找功能可以使你的工作更轻松。你可以通过单击列标题实现按字母顺序对这些列进行排序。如图 1-26 所示，此窗口显示了按名称排序的变量，你可以在标有“查找列名”的框中键入列名称，从而查找该列。

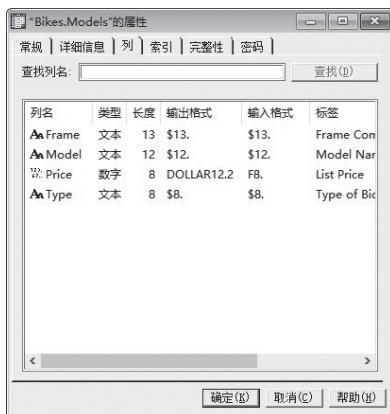


图 1-26

1.13 使用 SAS 系统选项

系统选项是你可以更改从而影响 SAS 的一些参数，包括 SAS 的工作方式、输出样式、使用的内存量、错误处理方式，等等。SAS 对你希望如何工作做了许多预设。这很好，你肯定不希望每次使用 SAS 时都去指定每一个小细节。但是，你可能并不总是喜欢 SAS 做出的预设。系统选项为你提供了更改某些预设的方法。

并非每个选项都适用于所有操作环境。SAS 帮助文档中将显示你的操作环境特定的选项列表。你可以通过打开“SAS 系统选项”窗口或使用 OPTIONS 过程来查看系统选项及其当前值的列表。要使用 OPTIONS 过程，请提交以下 SAS 程序并在 SAS 日志中查看结果：

```
PROC OPTIONS;
RUN;
```

指定系统选项的方式有四种，有些选项只能用这些方法中一部分来指定。你的操作环境的 SAS 帮助文档会说明每个系统选项的适用方法。

- (1) 创建包含系统选项设置的 SAS 配置文件。SAS 每次启动时都会访问该文件。配置文件由系统管理员创建。（如果你使用的是个人电脑，系统管理员可能就是你自己）。
- (2) 启动 SAS 时从系统提示符指定系统选项（称为调用）。
- (3) 如果使用 SAS 窗口环境，更改“SAS 系统选项”窗口中的所选选项。

(4) 在 SAS 程序中使用 OPTIONS 语句。

以上方法按优先级由低到高的顺序列出；方法 2 将覆盖方法 1，方法 3 将覆盖方法 2，依此类推。如果你在使用 SAS 窗口环境，方法 3 和方法 4（“SAS 系统选项”窗口和 OPTIONS 语句）将相互覆盖。因此，最后使用的那个方法将生效。本章节仅介绍最后两种方法。前两种方法非常依赖于系统。要了解有关这些方法的更多信息，请参阅有关你的操作环境的 SAS 帮助文档。

OPTIONS 语句 OPTIONS 语句是 SAS 程序的一部分，并影响其后的所有步。它以关键字 OPTIONS 开头，后面紧跟选项列表及其值，例如：

```
OPTIONS LEFTMARGIN = 1IN NODATE;
```

OPTIONS 语句是不属于 PROC 或 DATA 步的特殊 SAS 语句之一。这个全局语句可以出现在 SAS 程序中的任何地方，但通常最合理的方式是将其置于程序的第一行。这样你就可以很容易地看到哪些选项在起作用。如果 OPTIONS 语句处于 DATA 或 PROC 步中，则会影响该步以及后续步。程序中任何后续的 OPTIONS 语句都会覆盖先前的 OPTIONS 语句。

“SAS 系统选项”窗口 你可以通过“SAS 系统选项”窗口查看和更改 SAS 系统选项。通过在屏幕的命令行区域中键入 OPTIONS 或从菜单栏中选择工具>选项>系统来打开该窗口，如图 1-27 所示。要更改选项的值，请先通过单击屏幕左侧的相应类别来定位该选项。选项列表及其当前值将显示在屏幕的右侧。右键单击选项本身可以修改值或将其设置为默认值。



图 1-27

打印结果的选项 以下是你可能想要使用的一些系统选项，这些选项会影响打印格式（换言之，即非 HTML）的结果外观：

- CENTER | NOCENTER ● 控制输出结果是居中还是左对齐， 默认值为 CENTER；
- DATE | NODATE ● 控制是否在每页输出结果顶部显示当前日期， 默认值为 DATE；
- NUMBER | NONUMBER ● 控制是否在 SAS 输出的每一页显示页码， 默认值为 NUMBER；
- ORIENTATION = *orientation* ● 指定打印输出的方向， LANDSCAPE 还是 PORTRAIT， 默认值为 PORTRAIT；
- PAGENO = *n* ● 从 *n* 开始对输出页进行编号， 默认值为 1；
- RIGHTMARGIN = *n* ● 指定打印输出时的页边距（例如， 0.75 in 或 2cm）。默认值为 0.00 in。
- LEFTMARGIN = *n*
- TOPMARGIN = *n*
- BOTTOMMARGIN = *n*

2

第 2 章 导入数据到 SAS

实践是最好的导师。^①

——菲尔加拉赫

我们都从不断地做，不断地实验（甚至经常失败）和不断地发问中学习。^②

——杰·雅各布·温迪

① 出自《巴特利特引用词典》第 13 版，作者约翰·巴特利特，小布朗公司版权所有 1955 年。

② 来自 SAS L Listserv，1994 年 3 月 15 日。经作者许可转载。

2.1 导入数据到 SAS 的方法



数据以多种多样的形式存在。数据可能书写在纸张上或者输入计算机中的原始数据文件中。你的数据可能在个人计算机的数据库文件中，或者在办公室里大型机的数据库管理系统（DBMS）中。无论你的数据在哪里，SAS 总有办法使用它们。你可能需要转换数据格式，或者 SAS 能以数据当前的格式直接使用它们。本节概述了一些将数据导入 SAS 的不同方法。大多数的导入方法涵盖在本书之中，但是仍有少数更高级的方法仅仅只是简单提及，以便于让读者知道它们的存在。我们没有试图涵盖所有的导入方法，因为新的方法层出不穷。充满创造力的 SAS 用户总能想出适用于其特定情况的更聪明的办法。不管怎样，本书介绍的方法至少有一种会适合你。

数据导入 SAS 的方法可以被分为四个基本类别：

- 直接将数据输入SAS数据集
- 利用原始数据文件创建SAS数据集
- 将其他软件的数据文件转换成SAS数据集
- 直接读取其他软件的数据文件

当然，选择什么方法取决于你的数据在哪里，以及你手边有哪些工具软件可以使用。

直接将数据输入 SAS 数据集 有时候，将数据输入 SAS 最好的办法就是将数据通过键盘直接输入 SAS 数据集。

- Base SAS软件包含了一个VIEWTABLE窗口（将在第2.2节讨论）。VIEWTABLE能够让你以表格形式输入数据，你可以定义变量或者列，并为它们设置属性，例如：名称、长度、类型（字符或数值）。
- 在Base SAS的Windows版软件里还包含了SAS Enterprise Guide。它有一个类似于VIEWTABLE的数据输入窗口。如同使用VIEWTABLE一样，你能够定义变量以及给变量设置属性。
- SAS/FSP使你能够设计自定义的数据输入界面。当数据输入错误时，它还具备检测到这些错误的能力。SAS/FSP的许可不包含在Base SAS软件中。

从原始数据文件创建 SAS 数据集 本章的大部分内容都着重于描述读取原始数据文件（也称为文本文件、ASCII 文件、顺序文件或者平面文件）。由于 DATA 步是 Base SAS 软件的重要组成部分，所以你可以随时用它来读取原始数据文件。如果你的数据尚

不在原始数据文件中，你可以先将你的数据转换成原始数据文件。下面是两种读取原始数据文件的基本方法：

- DATA步（将从本章的2.4节中介绍）的功能如此强大以至于能读取几乎所有类型的原始数据文件。
- 导入向导（将从第2.3节中介绍）以及具有类似功能的IMPORT过程（将在第2.16节中介绍）都可以在UNIX和WINDOWS操作环境中使用。它们为读取特定类型的原始数据文件提供了简单的方法，包括逗号分隔文件（CSV）和其他分隔文件。

将其他软件的数据文件转换成SAS数据集 每个软件应用程序都有自己的数据文件格式。这对于软件开发者来说是非常有用的，但是对于软件使用者来说却非常麻烦，尤其是对于数据的存储和分析在不同的应用程序中的情况。以下是一些可用于数据转换的选项：

- 在UNIX和WINDOWS操作环境中，可以使用IMPORT过程和导入向导将Microsoft Excel、Lotus、dBase、Stata、SPSS、JMP、Paradox和Microsoft Access文件转换为SAS数据集。除了JMP以外，所有上述数据格式都需要你在计算机上安装SAS/ACCESS Interface to PC Files模块。导入向导将在第2.3节中介绍，使用IMPORT过程读取Excel文件将在第2.17节中介绍。
- 如果你没有SAS/ACCESS软件，你可以在你的应用程序中创建原始数据文件，然后通过DATA步或IMPORT过程读取它们。许多应用程序都能创建CSV文件，用导入向导、IMPORT过程（将在第2.3节和2.16节中介绍）或者DATA步（将在第2.15节中介绍）都能很容易地读取这种文件。
- 动态数据交换（DDE）仅可在Windows操作环境中使用。要使用DDE，其他的Windows应用程序（例如Microsoft Excel）需要和SAS同时运行在你的计算机上。然后，使用DDE和DATA步可以将数据转换为SAS数据集。

直接读取其他软件的数据文件 在某些特定情况下，你也许能够直接读取数据文件，而无须先将它转成SAS数据集。当有许多人一起更新数据文件的时候，这个方法尤为有用，这样可以确保你使用的是最新的数据。

- SAS/ACCESS产品使你能够直接读取数据文件，而无须预先将其转换成SAS数据集。SAS为大多数主流数据库管理系统提供了相应的SAS/ACCESS产品，其中包括：ORACLE、DB2、INGERS、MYSQL以及SYBASE。本书并不涵盖这种数据访问方法。

- 我们之前提到过，可以使用SAS/ACCESS Interface to PC Files将不同类型的个人计算机文件转换成SAS数据集。但是，你也能使用Excel、Access和JMP引擎直接读取这些类型的文件，无须做数据转换。关于这些引擎的详细信息，请参阅SAS帮助文档。
- Base SAS也包含了一些用于直接读取数据的引擎，包括针对于SPSS、OSIRIS、旧版本的SAS数据集以及SAS数据集传输格式的引擎。有关可用引擎的完整列表，请参见适用于你的操作环境的SAS帮助文档。

在以上这些数据导入方法中，你一定能找到一个或多个适用于你的方法。

2.2 使用 VIEWTABLE 窗口输入数据

VIEWTABLE 窗口，作为 Base SAS 软件^①的组成部分，提供了一种简易的方式来创建、浏览和编辑 SAS 数据集。从名称就能看出来，VIEWTABLE 窗口以表格形式显示表（数据集的另一个名称）。要打开 VIEWTABLE 窗口，从菜单栏中选择工具 ▶ 表编辑器，一个空的 VIEWTABLE 窗口随即打开，如图 2-1 所示。

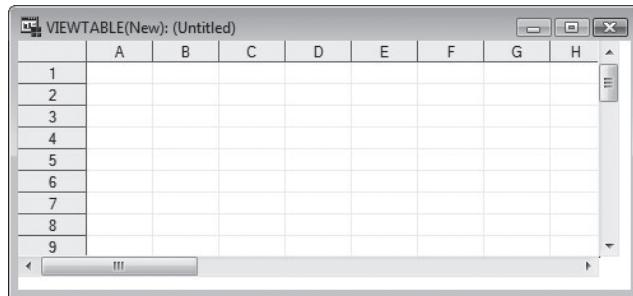


图 2-1

该表没有数据。你能看到以数字作为标签的行(观测)和以字母作为标签的列(变量)。你可以开始向这个默认表输入数据，SAS 将自动识别这个列是数值型还是字符型。然而，最好还是直接告知 SAS 数据的类型，以便每个列都按照你设想的方式建立。你可以通过“列属性”窗口来设置。

^① 如果你正在使用一个非图形化监控器，SAS 将用 FSWIEW 来展示你的表格。因此你还额外需要 SAS/FSP 软件的许可。

“列属性”窗口 列顶端的字母被作为默认的变量名称。右键单击一个字母，你可以选择打开该列的“列属性”窗口，如图 2-2 所示。该窗口包含了一系列默认值，你可以根据需求替换这些值。如果你想输入日期，你需要选择一个日期输入格式，以便输入的日期自动转换成 SAS 日期值^①。关于输入格式的详细信息，请参见第 2.7 节和第 2.8 节。如果你也选择了一个日期输出格式，它将被显示成一个可读的日期。有关输出格式的详细信息，请参见 4.6 节和 4.7 节。当你对这些值确定后，单击“应用”。要切换到其他列，在 VIEWTABLE 窗口里单击该列即可。列属性更改完成后，单击“关闭”按钮。

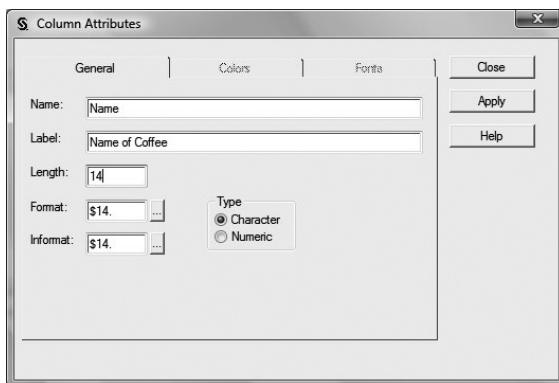


图 2-2

输入数据 列定义完成后，你就可以输入数据。要移动光标，可以单击一个字段或者使用制表符和箭头键。图 2-3 是一个已经定义完列属性并且输入了数据的表。

	Name of Coffee	Where Grown	Price
1	AA	Kenya	\$8.95
2	Antigua	Guatemala	\$9.95
3	Blue	Jamaica	\$9.95
4	Harrar	Ethiopia	\$8.95
5	Kaanapali Moka	Maui USA	\$16.95
6	Kona	Hawaii USA	\$18.95
7	Malulani	Molokai USA	\$15.95
8	Mocha Pure	Yemen	\$10.95
9	Santos	Brazil	\$9.95
10	Supremo	Columbia	\$10.95

图 2-3

保存表 要保存表，从菜单栏中选择文件 ▶ 另存为…打开一个“另存为”窗口（类

^① SAS 日期值是从 1960 年 1 月 1 日到该日期的天数。

似于图 2-4 所示的“打开”窗口。显示的逻辑库对应于计算机上的位置（例如目录），如果要将你的表保存到其他位置，你可以单击“新建逻辑库”图标，打开“新建逻辑库”窗口（如第 1.10 节和第 2.18 节所示），从而添加其他的逻辑库。在“另存为”窗口中，为你的表指定一个名称，然后单击“保存”。



图 2-4

打开现有的表 要浏览或编辑现有的表，从菜单栏中选择工具 ▶ 表编辑器打开 VIEWTABLE 窗口，然后从菜单栏中选择文件 ▶ 打开。在“打开”窗口中，单击你所需的逻辑库和表名，然后单击“打开”。如果所需的表格不在任何现有的逻辑库中，单击“创建新逻辑库”图标创建一个逻辑库。要从浏览模式（默认）切换到编辑模式，从菜单栏中选择编辑 ▶ 编辑模式。你也可以通过在 SAS 资源管理器窗口中导航到现有的表，通过双击打开它。

其他特性 VIEWTABLE 窗口还有许多其他的特性，包括排序、打印、添加和删除行以及查看多个行（默认设置，称作“表视图”）或者一次只查看一行（称作“表单视图”）。你可以使用图标或者菜单控制这些特性。

在 SAS 程序中使用表 在 VIEWTABLE 中创建的表可以在 SAS 程序中使用，正如 SAS 程序中创建的表可以在 VIEWTABLE 中使用一样。例如，如果你把一个名为 COFFEE 的表存在 SASUSER 逻辑库中，你可以用下面的程序将其打印出来：

```
PROC PRINT DATA = Sasuser.coffee;
RUN;
```

2.3 使用导入向导读取文件

在 Windows 和 UNIX 操作环境中使用导入向导，只需回答几个问题，就能将各种

各样的数据文件类型转换成 SAS 数据集。导入向导通过扫描你的文件来决定变量的类型^①，然后默认用数据的第一行作为变量名称。导入向导可以读取所有类型的分隔文件，包括逗号分隔值（CSV）文件，它是用于在应用程序之间移动数据的常用文件类型。此外，如果你有 SAS/ACCESS Interface to PC Files，则你还可以读取许多主流的 PC 文件类型。

从菜单栏中选择文件▶导入数据启动导入向导。从标准数据源列表中选择你要导入的文件类型，如图 2-5 所示。在本示例中，要导入的数据文件是逗号分隔值（CSV）文件。

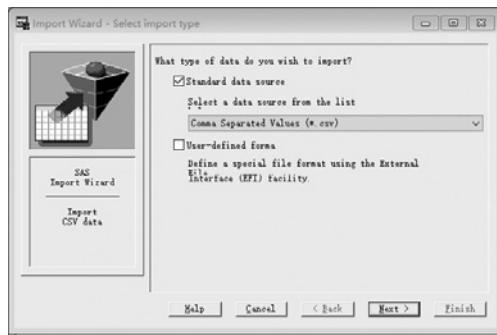


图 2-5

接下来，指明你要导入的文件所在的位置，如图 2-6 所示。默认情况下，SAS 使用文件的第一行作为 SAS 数据集的变量名称，并从第二行开始读取数据。当读取分隔文件、CSV 或者制表符分隔文件时，单击选项…按钮打开“分隔文件选项”窗口。

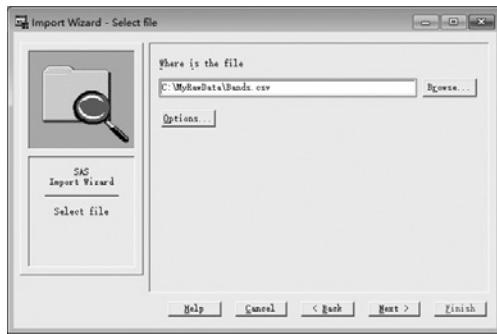


图 2-6

① 默认情况下，导入向导将扫描带分隔符的文件（delimited file）的前 20 行记录，Microsoft Excel 文件的前 8 行记录。如果某些行数据整体缺失，或者数据不能代表整个文件，导入向导（以及 IMPORT 过程步）可能无法正确地读取文件。更多信息参见第 2.16 节和 2.17 节。

对于分隔文件，在“分隔文件选项”窗口的“分隔符”框中指定分隔符，如图 2-7 所示。对于 CSV 或者制表符分隔文件，分隔符已经确定，所以窗口中该部分为灰色（无法修改）。在这个窗口里，你还能选择是否读取变量名称，指定从第几行开始读取数据，并设置用于猜测变量类型的数据行数。



图 2-7

如图 2-8 所示，下一个窗口要求你为要创建的 SAS 数据集选择 SAS 逻辑库和成员名称。如果你选择 WORK 逻辑库，则当你退出 SAS 的时候，该 SAS 数据集将被删除。如果你选择其他逻辑库，即使你退出 SAS，该 SAS 数据集仍能保留下。在导入向导中无法定义逻辑库，所以确保在进入导入向导之前，已经预先定义逻辑库。你可以使用第 1.10 节中介绍的“新建逻辑库”窗口定义逻辑库（或者使用 LIBNAME 语句，将在第 2.19 节中介绍）。在选择逻辑库之后，为 SAS 数据集输入一个成员名称。

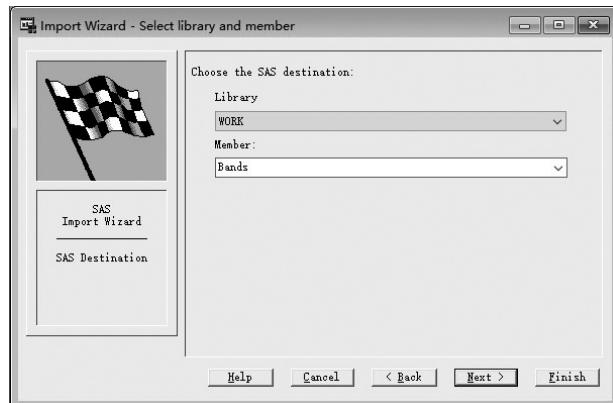


图 2-8

如图 2-9 所示，在最后一个窗口中，你可以选择是否将用于导入该文件的 PROC IMPORT 语句保存起来。

对于某些类型的文件，导入向导会问一些额外的问题。例如，如果你要导入 Microsoft Access 文件，将会询问你数据库名称和要导入的表的名称。

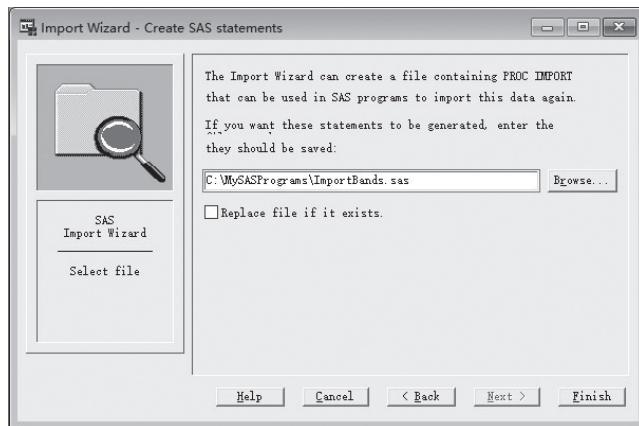


图 2-9

在 SAS 程序中使用导入的数据 通过导入向导导入的数据可以用在任何 SAS 程序中。例如，如果你把数据保存在 WORK 逻辑库中，并且命名为 BANDS，你可以用以下程序将其打印出来：

```
PROC PRINT DATA = WORK.bands;  
RUN;
```

或者，由于 WORK 是默认的逻辑库，你也可以使用以下程序：

```
PROC PRINT DATA = bands;  
RUN;
```

2.4 指定原始数据位置

如果你的数据存储在原始数据文件（也被称为文本文件、ASCII 文件、顺序文件或者平面文件）中，DATA 步为你提供了最大程度的灵活性来读取这些文件。读取原始数据文件的第一步是要告知 SAS 原始数据的位置。你的原始数据可能在你 SAS 程序内部（也

被称作“instream”），或者在一个单独的文件里。不管是哪种方式，你必须告知 SAS 这些数据的位置。

原始数据文件可以用简单的文本编辑器或者系统命令来查看。对于个人计算机用户来说，原始数据文件要么不和任何应用程序关联，要么与一个简单的编辑器相关联，例如 Microsoft 记事本。在一些操作环境中，你可以使用命令列出这些文件。例如，在 UNIX 上使用 CAT 或者 MORE 命令。某些数据文件并非原始数据，例如电子表格。如果你试图用文本编辑器来查看电子表格文件，你很可能看到很多键盘上找不到的有趣字符。它可能会导致你的计算机发出警报声，你会因此希望自己躲在走廊尽头的那间私人办公室。它看起来一点都不像当你使用电子表格编辑软件打开时那样的优美和整洁。

内部原始数据 如果直接在你的 SAS 程序里输入原始数据，这些数据将成为程序的内部数据。当数据量很小或者你准备使用小的测试数据集测试一段程序的时候，你很可能要这么做。你可以使用 DATALINES 语句表明内部数据。DATALINES 语句必须是 DATA 步中最后一条语句。SAS 程序中 DATALINES 语句后的所有行都将被视为数据，直到 SAS 遇到一个分号为止。分号可以自己独占一行，也可以放在数据行的语句结尾处。在数据之后的任何语句都将成为新的程序步的一部分。如果你年龄足够大，还记得计算机穿孔卡（punching computer cards），你可能会喜欢用 CARDS 语句来代替 DATALINES。CARDS 语句和 DATALINES 语句的作用是一样的。下面这段 SAS 程序展示了如何使用 DATALINES 语句。（DATA 语句只是告知 SAS 创建一个名为 USPERSIDENTS 的 SAS 数据集，而 INPUT 语句告知 SAS 如何读取该数据。INPUT 语句将在第 2.5~2.15 节中介绍）。

```
*将内部数据读取到SAS数据集uspresidents;
DATA uspresidents;
  INPUT President $ Party $ Number;
  DATALINES;
Adams F 2
Lincoln R 16
Grant R 18
Kennedy D 35
;
RUN;
```

外部原始数据文件 你通常会把数据保存在外部文件中，从而将数据和程序分离。这消除了你在编辑 SAS 程序时意外篡改数据的可能性。使用 INFILE 语句告知 SAS 文件名和路径，以及从外部文件读取数据的选项。INFILE 语句紧随在 DATA 语句之后，并且

必须在 INPUT 语句之前。在 INFILE 关键字之后是包含在引号中的文件路径和名称。下面是一些来自于不同操作环境的例子：

```
Windows:      INFILE 'c:\MyDir\President.dat';
UNIX:        INFILE '/home/mydir/president.dat';
z/OS:         INFILE 'MYID.PRESIDEN.DAT';
```

假设下面的数据来自于 C 盘（Windows 系统）MyRawData 目录下名为 President.dat 的数据文件：

```
Adams F 2
Lincoln R 16
Grant R 18
Kennedy D 35
```

下面的程序展示了如何使用 INFILE 语句读取该外部数据文件：

```
*从外部文件读取数据到SAS数据集;
DATA uspresidents;
  INFILE 'c:\MyRawData\President.dat';
  INPUT President $ Party $ Number;
RUN;
```

SAS 日志 无论任何时候，当你从外部数据文件中读取数据时，SAS 都将在日志文件中记录一些关于该文件非常有价值的信息。下面摘录了运行上面 SAS 代码所产生的 SAS 日志。在你读取一个文件后，请务必检查这个信息，因为它很可能指明了问题的所在。通过简单地比较从输入文件中读取的记录行数和 SAS 数据集中写入的观测数，就能知道 SAS 是否正确地读取了数据。

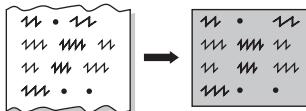
```
NOTE: INFILE 'c:\MyRawData\President.dat' 是:
      文件名=c:\MyRawData\President.dat,
      RECFM=V, LRECL=256
NOTE: 从 INFILE 'c:\MyRawData\President.dat' 中读取了 4 条记录。
      最小记录长度是 9。
      最大记录长度是 12。
NOTE: 数据集 WORK.USPRESIDENTS 有 4 个观测和 3 个变量。
```

长记录 在一些操作环境中，SAS 假设外部文件中的记录长度不超过 256 个字符（记录长度为一行数据的字符个数，其中包括了空格）。如果你的记录很长，而且 SAS 似乎没有读取所有数据，你可以在 INFILE 语句中使用“LRECL=” 选项，指定一个至少和数据文件中最长记录等长的记录长度。

```
INFILE 'c:\MyRawData\President.dat' LRECL=2000;
```

请查看 SAS 日志，以确保最大记录长度和你设想的一样。

2.5 读取空格分隔的原始数据



如果你的原始数据文件中的所有值之间都被至少一个空格^①分隔开，合适的方法是使用列表输入（也被称作自由格式输入）读取该数据。列表输入是将原始数据读取到 SAS 中的简单方法，简便的同时也有一些局限。你必须读取一条记录中的全部数据，不能跳过不需要的值。任何缺失值都必须用句点来标识。如果有字符型数据，则它必须足够简单：没有内嵌的空格，长度不超过 8 个字符^②。如果数据文件中包含日期或者其他需要特殊处理的数据，列表输入可能就不太合适。听起来它有很多的限制，但是事实上通过列表输入读取的数据文件数量是惊人的。

INPUT 语句，作为 DATA 步的组成部分，告知 SAS 如何读取原始数据。要编写使用列表输入的 INPUT 语句，只需在 INPUT 关键字之后按照变量名称在数据文件中出现的顺序列出它们即可。一般来说，变量名称不能超过 32 个字符，以字母或者下划线开头，且只包含字母、下划线和数字。如果变量值为字符（非数值）类型，在变量名称之后放置一个美元符号 (\$)。在变量名称之间至少留一个空格，记得在语句的结尾处放置一个分号。下面是一个简单的列表样式的 INPUT 语句的示例：

```
INPUT Name $ Age Height;
```

该语句告知 SAS 读取三个数据值。Name 之后的美元符号 (\$) 表明该变量为字符型，而 Age 和 Height 变量都是数值型。

示例 你的家乡今年蟾蜍泛滥成灾。一个本地居民，在听说了加利福尼亚州的青蛙跳跃比赛之后，萌生了一个想法：组织一场蟾蜍跳跃比赛作为该年度城镇集会的压轴节目。在每场竞赛当中，蟾蜍都有自己的名称、体重和三次尝试跳跃距离。如果蟾蜍的某次跳跃犯规，则用一个句点标识缺失数据。下面就是这个数据文件 ToadJump.dat：

① SAS 能使用列表输入（list input）读取包含其他分隔符的文件，例如逗号或者制表符。参见第 2.15 节和 2.16 节。

② 使用 LENGTH 语句来覆盖这个限制是可行的（将在第 11.12 节讨论），它会将字符变量的长度从默认的 8 变成 1 ~ 32767 之间的任何值。

```
Lucky 2.3 1.9 . 3.0  
Spot 4.6 2.5 3.1 .5  
Tubs 7.1 . . 3.8  
Hop 4.5 3.2 1.9 2.6  
Noisy 3.8 1.3 1.8  
1.5  
Winner 5.7 . . .
```

这个数据文件看起来不是很整洁，但是它确实满足了列表输入所有要求：字符数据长度不超过 8 个字符且没有内嵌空格，所有的数值被至少一个空格分隔，缺失数据用一个句点标识。注意，Noisy 的数据溢出到了下一行。这并没有问题，因为如果 INPUT 语句中的变量个数比一行数据中的值的数量多，SAS 将默认到下一行读取。

下面是用来读取这个数据文件的 SAS 程序：

```
* 创建一个名字为 toads 的 SAS 数据集；  
* 使用列表输入，读取数据文件 ToadJump.dat；  
DATA toads;  
    INFILE 'c:\MyRawData\ToadJump.dat';  
    INPUT ToadName $ Weight Jump1 Jump2 Jump3;  
RUN;  
* 打印数据以确保文件被正确读取；  
PROC PRINT DATA = toads;  
    TITLE 'SAS Data Set Toads';  
RUN;
```

变量 ToadName、Weight、Jump1、Jump2 和 Jump3 按照数据文件中相同的顺序被列在了 INPUT 关键字之后。ToadName 后面的美元符号 (\$) 说明它是一个字符型变量，其他的所有变量都为数值型。使用 PROC PRINT 语句打印读取后的数值以确保它们正确。PRINT 过程以最简单的方式打印了 SAS 数据集中所有变量的值和所有观测。PROC PRINT 后面的 TITLE 语句告知 SAS 把引号内的文本放在每个输出页面的顶端。如果你的程序里没有 TITLE 语句，SAS 将把 “The SAS System” 放在每个页面的顶端。

表 2-1 是 PRINT 过程的输出结果。一件重要的事情是，请经常检查你所创建的数据集以确保它们正确。你也可以使用 VIEWTABLE 窗口（在第 1.11 节中介绍）查看该数据。

表 2-1 SAS Data Set Toads

Obs	ToadName	Weight	Jump1	Jump2	Jump3
1	Lucky	2.3	1.9	.	3
2	Spot	4.6	2.5	3.1	0.5

(续表)

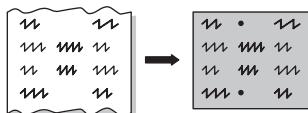
Obs	ToadName	Weight	Jump1	Jump2	Jump3
3	Tubs	7.1	.	.	3.8
4	Hop	4.5	3.2	1.9	2.6
5	Noisy	3.8	1.3	1.8	1.5
6	Winner	5.7	.	.	.

由于 SAS 需要到下一行去获取 Noisy 最终一跳的数据，SAS 日志中出现了下面的提示：

NOTE: INPUT 语句到达一行的末尾, SAS 已转到新的一行

如果在你在意料之外看到这条注释，则可能意味着出了问题。如果是这样，请参见详细介绍该注释的第 11.4 节。

2.6 读取按列排列的原始数据



一些数据文件在所有值或表示缺失值的句点之间没有空格（或者其他分隔符），因此无法使用列表输入来读取该文件。

但是，如果每个变量值都能在数据行的相同位置找到，只要所有值都是字符型或者标准数值型，你就可以使用列输入来读取。标准数值数据仅包含数字、小数点、正负号和代表科学计数法的 E。例如，带有内嵌逗号和日期的数值就不是标准数值类型。

列输入对比于列表输入具有如下的一些优势：

- 值之间无须空格
- 缺失值可以留空
- 字符型数据可以内嵌空格
- 可以跳过不需要的变量

调查数据非常适合使用列输入进行读取。调查问卷的答案经常以个位数字（0 ~ 9）编码。如果在每个值之间插入一个空格，这个文件大小将翻倍，并且相比于没有空格的文件，需要输入两倍的数据量。带有街道地址的数据集通常需要内嵌空格，这点也非常适合列输入。街道 Martin Luther King Jr. Boulevard 应该被读取为一个变量，如果使用列表输入，它将被错误地读取成五个变量。能被列输入读取的数据常常也能被格式化输入

或者多种输入方式的组合读取（在第2.7节、第2.8节和第2.9节中介绍）。

在使用列输入时，INPUT语句以下面这种方式构成：在INPUT语句之后，列出第一个变量名称。如果变量是字符类型，则留一个空格，后面紧跟着一个美元符号 (\$)。在美元符号 (\$) 或者变量名称（如果是数值类型）之后再留一个空格，然后列出该变量对应的列或列范围。这些列表示数值或者字符在数据行中的位置，请不要和你在电子表格中看到的列混淆。为所有要读取的变量重复以上过程。下面是使用列样式的简单 INPUT 语句：

```
INPUT Name $ 1-10 Age 11-13 Height 14-18;
```

第一个变量 Name 是一个字符型变量，它的数据值在第 1 ~ 10 列。Age 和 Height 变量后没有美元符号 (\$)，因此它们是数值型变量，这两个变量的数据值在其名称之后列出的列范围内。

示例 当地的小联盟棒球队 Walla Walla Sweets 保持着球场内的特许销售记录。球场里最受欢迎的商品是甜洋葱圈，小卖部和露天看台上的商贩都在卖。球场的主人感觉如果一场比赛有许多击打和奔跑，露天看台上会比小卖部卖更多的洋葱圈。他们认为可以在比赛升温的时候让更多的商贩去露天看台上卖。在此之前，他们需要一些证据来支持他们的想法。

对于每个主场比赛，他们有如下的信息：对方球队的名称，小卖部和露天看台分别卖了多少洋葱圈，每队击打球的次数以及每队的最终得分。下面的例子是一个名为 OnionRing.dat 的数据文件。为了查看方便，在数据顶部放置一个显示列编号的列标尺。

	1	2	3	4
Columbia Peaches	35	67	1	10
Plains Peanuts	210		2	5
Gilroy Garlices	151035	12	11	7
Sacramento Tomatoes	124	85	15	4

请注意，该数据文件有如下特征，这些特征使列输入成为一个首要的选择：所有值都以列的方式排列，球队名称有内嵌的空格，缺失值留空，以及其中有一处的数值之间没有空格（Gilroy Garlices 的球迷一定非常热爱洋葱圈）。

下面这段代码向你展示了如何使用列输入来读取该数据文件。

```
*创建一个名字为sales的SAS数据集；  
*使用列输入，读取数据文件OnionRing.dat；
```

```

DATA sales;
  INFILE 'c:\MyRawData\OnionRing.dat';
  INPUT VisitingTeam $ 1-20 ConcessionSales 21-24 BleacherSales 25-28
        OurHits 29-31 TheirHits 32-34 OurRuns 35-37 TheirRuns 38-40;
RUN;
*打印数据以确保文件被正确读取;
PROC PRINT DATA = sales;
  TITLE 'SAS Data Set Sales';
RUN;

```

变量 VisitingTeam 为字符型（由 \$ 符号标识），从文件的第 1 ~ 20 列读取客队名称。变量 ConcessionSales 和 BleacherSales 分别从第 21 ~ 24 列和第 25 ~ 28 列读取小卖部和露天看台的销量数据。主队击打次数 OurHits 和客队击打次数 TheirHits 分别从第 29 ~ 31 列和第 32 ~ 34 列中读取。主队奔跑次数 OurRuns 从第 35 ~ 37 列中读取，而客队奔跑次数 TheirRuns 则从第 38 ~ 40 列中读取。

下面是 PRINT 过程输出的结果，你也可以使用 VIEWTABLE 查看该数据。

表 2-2 销售数据集

Obs	VisitingTeam	ConcessionSales	BleacherSales	OurHits	TheirHits	OurRuns	TheirRuns
1	Columbia Peaches	35	67	1	10	2	1
2	Plains Peanuts	210	.	2	5	0	2
3	Gilroy Garlics	15	1035	12	11	7	6
4	Sacramento Tomatoes	124	85	15	4	9	1

2.7 读取非标准格式的原始数据



有的时候，原始数据并非直接是数值或者字符。例如，我们人类很容易将数值 1,000,001 读作“一百万零一”，但是你所信任的计算机却只能将其读成字符。

数值内嵌的逗号方便我们解读，但是对于计算机而言，在没有指令的情况下不可能识别内嵌逗号的数值。在 SAS 中，使用输入格式告知计算机如何解释这些类型的数据。

任何时候，当你读取非标准数据时，输入格式都很有用（标准数值数据只包含数字、小数点、正负符号和科学记数法 E）。嵌入逗号或美元符号的数值就是非标准数据的例子。其他例子包括十六进制数据和压缩十进制格式数据。SAS 也有读取这些类型的数据的输入格式。

日期^①可能是最常见的非标准数据。SAS可以使用日期输入格式将惯用的日期格式，例如10-31-2013或者31OCT13转成一个数值，即从1960年1月1日到该日期的天数。该数值被称为SAS日期值（为什么是1960年1月1日？不太清楚，也许1960年是SAS创始人的幸运年）。当你要使用日期进行计算的时候，这样做非常有用。例如，你能通过将两个日期相减，非常容易地算出它们之间相隔的天数。

输入格式的三个基本类型是：字符、数值和日期。下一节中可以看到一个表格列出了SAS里可选择的输入格式。这三种类型都有如下的一般形式：

字符	数值	日期
<i>\$informatw.</i>	<i>informatw.d</i>	<i>informatw.</i>

美元符号(\$)表明这是字符型的输入格式，*informat*是该输入格式的名称，*w*是总长度，*d*是小数位数（仅限数值型输入格式）。句点是输入格式名称里非常重要的组成部分。没有句点，SAS将会把输入格式解释为变量名称，而变量名称默认情况下不能包含除下划线以外的任何特殊字符。有两个输入格式没有名称：*\$w.*用来读取标准的字符型数据，*w.d*用来读取标准的数值型数据。

要使用输入格式，在INPUT语句的变量名称之后放置输入格式，称为格式化输入。下面的INPUT语句就是格式化输入的示例：

```
INPUT Name $10. Age 3. Height 5.1 BirthDate MMDDYY10.;
```

为每个变量读取的列由起始点和输入格式的宽度所决定。SAS总是从第1列开始，因此第一个变量Name（其输入格式为\$10.）的数据值在第1~10列。第二个变量的起始点在第11列，SAS开始从第11~13列为Age读取值。第三个变量Height的值在第14~18列，这五个列包含了小数位和小数点本身（如150.3）。最后一个变量BirthDate的值从第19列开始，它是日期格式。

示例 下面这个示例展示了如何使用输入格式读取数据。下面的数据文件Pumpkin.dat表示了本地一场南瓜雕刻竞赛的结果。每一行数据都包含了竞赛者的姓名、年龄、类型（雕刻或者装饰）、南瓜的登记日期以及五个裁判分别的评分。

```
Alicia Grossman 13 c 10-28-2012 7.8 6.5 7.2 8.0 7.9
Matthew Lee 9 D 10-30-2012 6.5 5.9 6.8 6.0 8.1
Elizabeth Garcia 10 C 10-29-2012 8.9 7.9 8.5 9.0 8.8
```

^① 将在第3.8节中详细讨论在SAS中使用日期。

```
Lori Newcombe      6 D 10-30-2012 6.7 5.6 4.9 5.2 6.1
Jose Martinez     7 d 10-31-2012 8.9 9.5 10.0 9.7 9.0
Brian Williams    11 C 10-29-2012 7.8 8.4 8.5 7.9 8.0
```

下面这段程序用来读取这些数据。注意，有很多方法可以读取这些数据。如果你想到了其他的方法，也是可以的。

```
*创建一个名字为contest的SAS数据集;
*使用输入格式，读取数据文件Pumpkin.dat;
DATA contest;
  INFILE 'c:\MyRawData\Pumpkin.dat';
  INPUT Name $16. Age 3. +1 Type $1. +1 Date MMDDYY10.
        (Score1 Score2 Score3 Score4 Score5) (4.1);
RUN;
*打印数据以确保文件被正确读取;
PROC PRINT DATA = contest;
  TITLE 'Pumpkin Carving Contest';
RUN;
```

变量 Name 的输入格式为 \$16., 表示它是一个宽度为 16 个列的字符型变量。变量 Age 的输入格式为 3, 表示它是数值型变量, 宽度为 3 列, 没有小数位。+1 表示跳过一列。变量 Type 为字符类型, 宽度为 1 列。变量 Date 的输入格式为 MMDDYY10., 它能读取诸如 10-31-2013 或者 10/31/2013 格式的日期, 每种日期格式的宽度都为 10 列。其他的变量 Score1 到 Score5 都有相同的输入格式 4.1。通过把变量和输入格式分别放到小括号里, 你只需要列出一次该输入格式。

这是 PRINT 过程打印出来的结果, 你也可以使用 VIEWTABLE 查看该数据。

表 2-3 南瓜雕刻竞赛表

Obs	Name	Age	Type	Date ^①	Score1	Score2	Score3	Score4	Score5
1	Alicia Grossman	13	c	19294	7.8	6.5	7.2	8	7.9
2	Matthew Lee	9	D	19296	6.5	5.9	6.8	6	8.1
3	Elizabeth Garcia	10	C	19295	8.9	7.9	8.5	9	8.8
4	Lori Newcombe	6	D	19296	6.7	5.6	4.9	5.2	6.1
5	Jose Martinez	7	d	19297	8.9	9.5	10	9.7	9
6	Brian Williams	11	C	19295	7.8	8.4	8.5	7.9	8

① 注意这些日期被打印成了数字, 它是从 1960 年 1 月 1 日开始到那天为止的天数。第 4.6 节将讨论如何把这些值转换成可读的日期。

2.8 常用输入格式

表 2-4 常用输入格式^①

输入格式	定 义	宽度范围	默认宽度
字符类型			
\$CHARw.	读取字符数据——不会修剪前部和尾部的空白	1 ~ 32767	8 或者变量的长度
\$UPCASEw.	将字符数据转换成大写	1 ~ 32767	8
\$w.	读取字符数据，修剪前导空白	1 ~ 32767	无
日期、时间和日期时间类型^①			
ANYDTDTEw.	读取各种日期格式的日期	5 ~ 32	9
DATEw.	读取 <i>ddmmmyy</i> 或 <i>ddmmmyyyy</i> 格式的日期	7 ~ 32	7
DATETIMEw.	读取 <i>ddmmmyy hh:mm:ss.ss</i> 格式的日期时间	13 ~ 40	18
DDMMYYw.	读取 <i>ddmmyy</i> 或 <i>ddmmyyyy</i> 格式的日期	6 ~ 32	6
JULIANw.	读取 <i>yyddd</i> 或 <i>yyyyddd</i> 格式的儒略日期	5 ~ 32	5
MMDDYYw.	读取 <i>mmddyy</i> 或 <i>mmddyyyy</i> 格式的日期	6 ~ 32	6
STIMERw.	读取 <i>hh:mm:ss.ss</i> (或者 <i>mm:ss.ss</i> 或者 <i>ss.ss</i>) 格式的时间	1 ~ 32	10
TIMEw.	读取 <i>hh:mm:ss.ss</i> 或 <i>hh:mm</i> 格式的时间	5 ~ 32	8
数值类型			
COMMAw.d	删除内嵌的逗号和美元符号(\$), 将左括号转换成负号	1 ~ 32	1
COMMAXw.	就像 COMMAw.d 一样, 但是颠倒了逗号和句点的角色	1 ~ 32	1
PERCENTw.	把百分数转成数值	1 ~ 32	6
w.d	读取标准的数值数据	1 ~ 32	无

^① SAS 日期 (DATE) 值是从 1960 年 1 月 1 日到那天为止的天数, 时间 (TIME) 值是午夜之后到那个时间为止的秒数, 日期时间 (DATETIME) 值是从 1960 年 1 月 1 日午夜到那个时间为止的秒数。

(续表)

输入格式	输入数据	INPUT 语句	结果
字符类型			
\$CHARw.	my cat my cat	INPUT Animal \$CHAR10.;	my cat my cat
\$UPCASEw.	my cat	INPUT Name \$UPCASE10.;	MY CAT
\$w.	my cat my cat	INPUT Animal \$10.;	my cat my cat
日期、时间和日期时间			
ANYDTDTEw.	1jan1961 01/01/61	INPUT Day ANYDTDTE10.;	366 366
DATEw.	1jan1961 1 jan 61	INPUT Day DATE10.;	366 366
DATETIMEw.	1jan1960 10:30:15 1jan1961,10:30:15	INPUT Dt DATETIME18.;	37815 31660215
DDMMYYw.	01.01.61 02/01/61	INPUT Day DDMMYY8.;	366 367
JULIANw.	61001 1961001	INPUT Day JULIAN7.;	366 366
MMDDYYw.	01-01-61 01/01/61	INPUT Day MMDDYY8.;	366 366
STIMERw.	10:30 10:30:15	INPUT Time STIMER8.;	630 37815
TIMEw.	10:30 10:30:15	INPUT Time TIME8.;	37800 37815
数值类型			
COMMAw.d	\$1,000,001 (1,234)	INPUT Income COMMA10.;	1000001 -1234
COMMAXw.	\$1.000.001 (1.234,25)	INPUT Value COMMAX10.;	1000001 -1234.25
PERCENTw.	5% (20%)	INPUT Value PERCENT5.;	0.05 -0.2
w.d	1234 -12.3	INPUT Value 5.1;	123.4 -12.3

2.9 混合的输入样式

三种主要的输入样式每个都有自己的优势。列表样式最简单，列样式稍微难一点，而格式化样式是三种样式里最难的。然而，列样式和格式化样式不需要变量之间有空格（或者其他分隔符），还能读取内嵌的空白。格式化样式可以读取诸如日期的特殊数据。有时候你用一种样式，有时候用另外一种，还有时候最简单的方法是用多种样式的组合。SAS 灵活到你可以任意搭配输入样式从而方便你的输入。

示例 下面的原始数据包含美国国家公园的信息：名称、所在州名称（或者多个州名称）、建立的年份和占地面积（英亩）。

Yellowstone	ID/MT/WY	1872	4,065,493
Everglades	FL	1934	1,398,800
Yosemite	CA	1864	760,917
Great Smoky Mountains	NC/TN	1926	520,269
Wolf Trap Farm	VA	1966	130

你可以使用多种方式（本节的介绍要点）书写 INPUT 语句来读取该数据。下面的程序展示了其中的一种方式。

```
*创建一个名字为nationalparks的SAS数据集;
*使用混合输入方式, 读取数据文件NatPark.dat;
DATA nationalparks;
  INFILE 'c:\MyRawData\NatPark.dat';
  INPUT ParkName $ 1-22 State $ Year @40 Acreage COMMA9.;
RUN;
PROC PRINT DATA = nationalparks;
  TITLE 'Selected National Parks';
RUN;
```

注意，变量 ParkName 是用列样式输入进行读取，State 和 Year 则是用列表样式输入进行读取，而 Acreage 是用格式化样式输入进行读取。下面是 PRINT 过程的输出结果。你也可以用 VIEWTABLE 查看该数据。

表 2-5 美国国家公园表

Obs	ParkName	State	Year	Acreage
1	Yellowstone	ID/MT/WY	1872	4065493
2	Everglades	FL	1934	1398800
3	Yosemite	CA	1864	760917

(续表)

Obs	ParkName	State	Year	Acreage
4	Great Smoky Mountains	NC/TN	1926	520269
5	Wolf Trap Farm	VA	1966	130

有时候，程序员在混合使用输入样式时会遇到一些问题。当 SAS 读取一行原始数据的时候，它用一个指针来标记其位置，但是每种输入样式使用指针的方式有所不同。在使用列表样式输入时，SAS 自动扫描下一个非空的字段，然后开始读取。在使用列样式输入时，SAS 精确地读取你指定的列上的数据。但是在使用格式化样式输入时，SAS 只是开始读——不管指针现在哪里，SAS 都从那个位置继续读取。

有时候，你需要显式地移动指针。你可以使用列指针 @n 进行移动，其中 n 是 SAS 要移动到的列。

在前面的程序里，列指针 @40 告知 SAS 在读取数据值之前将指针移动到第 40 列。如果你像下面这条语句一样将 INPUT 语句中的列指针去掉，SAS 将从 Year 的后面紧接着读取 Acreage。

```
INPUT ParkName $ 1-22 State $ Year Acreage COMMA9.;
```

输出结果如表 2-6 所示。

表 2-6 美国国家公园

Obs	ParkName	State	Year	Acreage
1	Yellowstone	ID/MT/WY	1872	4065
2	Everglades	FL	1934	.
3	Yosemite	CA	1864	.
4	Great Smoky Mountains	NC/TN	1926	5
5	Wolf Trap Farm	VA	1966	.

因为 Acreage 是用格式化输入读取的，SAS 则从指针当前的位置开始读取。下面的数据文件顶部有用于列计数的列标尺，星号标识了 SAS 开始读取 Acreage 值的位置。

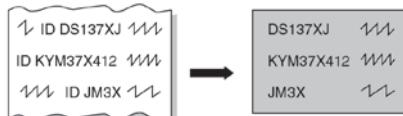
```
-----1-----2-----3-----4-----5
Yellowstone      ID/MT/WY 1872 * 4,065,493
Everglades       FL 1934 * 1,398,800
Yosemite        CA 1864 * 760,917
```

```
Great Smoky Mountains NC/TN 1926 *      520,269  
Wolf Trap Farm           VA 1966 *      130
```

名为 COMMA9. 的输入格式告知 SAS 读取 9 列，即便这些列全部为空白，SAS 也会读取。

列指针 @n 还有其他的用处。无论何时，当你希望在一行数据里向前或者向后移动列指针的时候，你都可以使用它。例如，你可以用它跳过不需要的数据，或者使用不同的输入格式两次读取同一个变量。

2.10 读取杂乱的原始数据



The diagram illustrates the transformation of raw, unstructured data into a clean dataset. On the left, a 'messy' input file is shown with three rows of data. The first row has fields 'ID DS137XJ' and '111'. The second row has 'ID KYM37X412' and '111'. The third row has '111 ID JM3X' and '111'. An arrow points from this to the right, where a 'clean' output file is shown with three columns: 'DS137XJ', '111', 'KYM37X412', '111', and 'JM3X', '111'.

有时候，你需要读取那种并没有排列整齐或者无法预知要读取的字符长度的数据。当你遇到这种杂乱的文件的时候，普通的列表输入、列输入或者格式化输入就不够用了。你的工具包里还需要有更多的工具：类似于 '@'character' 列指针或者冒号修饰符这样的工具。

@'character' 列指针 在第 2.9 节，我们向你展示了如何在读取数据之前使用 @ 列指针将指针移动到特定的列。然而，有时候你并不知道数据开始的位置，但是你知道它总是在一个特定的字符或者单词之后开始。在这种情况下，你可以使用 '@'character' 列指针。例如，假设你有一个关于狗类品种的数据文件。在该文件中，数据排列的并不整齐。但是你知道，狗的品种信息总是出现在单词 Breed 之后，你可以使用下面的 INPUT 语句读取狗的品种信息：

```
INPUT @'Breed:' DogBreed $;
```

冒号修饰符 只要狗的品种名称不超过 8 个字符（字符变量的默认长度），上面的 INPUT 语句就运转良好。因此，如果狗的品种是 Shepherd，这一切没有问题。但是，如果狗的品种是 Rottweiler，你只能得到 Rottweil。如果你在 INPUT 语句里给变量指定输入格式（例如 \$20.），告知 SAS 该变量的字段为 20 个字符的长度，SAS 将连续读取 20 个字符，而无论其中是否有空格^①。因此，DogBreed 变量可能会包含不需要的字符（在数

^① 使用 LENGTH 和输入格式语句代替 INPUT 为变量定义一个长度也是可以的。当一个变量的长度在 INPUT 语句之前定义的时候，SAS 将一直读取直到它遇到一个空格或者达到之前定义的长度为止。这和冒号修饰符的行为是一致的。输入格式语句在第 2.21 节讨论，LENGTH 语句在第 11.12 节讨论。

据行中狗品种后面的字符）。如果你只希望 SAS 读取到空格或者数据行的结束为止^①，那你在输入格式中使用冒号修饰符。要使用冒号修饰符，只需在输入格式前面放置一个冒号（:），例如使用 :\$20. 而不是 \$20..

例如，对于下面这行原始数据：

```
My dog Sam Breed: Rottweiler Vet Bills: $478
```

下面列举了使用不同 INPUT 语句所得到的结果：

语句	变量 DogBreed 的值
INPUT @'Breed: ' DogBreed \$;	Rottweil
INPUT @'Breed: ' DogBreed \$20.;	Rottweiler Vet Bill
INPUT @'Breed: ' DogBreed :\$20.;	Rottweiler

示例 每年，来自美国和加拿大的工程系学生都会建造混凝土轻舟，并参加地区和国家的竞赛。其中一部分竞赛项目是轻舟竞速。下面的数据包含了男子轻舟冲刺比赛的最终成绩。数据行里首先是轻舟的名称，然后是学校名称和时间。

```
Bellatorum School: CSULA Time: 1:40.5
The Kraken School: ASU Time: 1:45.35
Black Widow School: UoA Time: 1:33.7
Koicrete School: CSUF Time: 1:40.25
Khaos School: UNLV Time: 2:03.45
Max School: UCSD Time: 1:26.47
Hakuna Matata School: UCLA Time: 1:20.64
Prospector School: CPSLO Time: 1:12.08
Andromeda School: CPP Time: 1:25.1
Kekoapohaku School: UHM Time: 1:24.49
```

你可以看出来，因为轻舟的名称包含的字符个数是各不相同的，学校的名称也没有在一个列上排列整齐。此外，有的时间值是六个字符，有的是七个字符。下面这段程序只读取学校名称和时间。

```
DATA canoerresults;
  INFILE 'c:\MyRawData\Canoes.dat';
  INPUT @'School:' School $ @'Time:' RaceTime :STIMER8. ;
RUN;
PROC PRINT DATA = canoerresults;
  TITLE "Concrete Canoe Men's Sprint Results";
RUN;
```

^① 空格是默认的分隔符。这个方法也适用于其他的分隔符。更多关于读取分隔数据的信息请参见第 2.15 节和第 2.16 节。

该 INPUT 语句使用 '@'School:' 和 '@'Time:' 定位列指针，以便读取学校名称和时间。因为时间的字符数并不总是相同，所以使用带有冒号修饰符的输入格式 :STIMER8. 读取时间。如果没有冒号修饰符，在 SAS 读取到行末尾的时候，它将试图去下一行读取时间值。

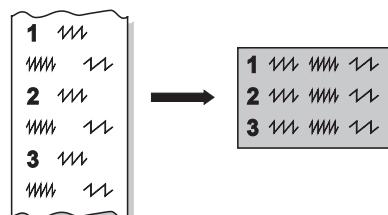
表 2-7 是 PRINT 过程的输出结果，你也可以使用 VIEWTABLE 查看该数据。

表 2-7 Concrete Canoe Men's Sprint Results

Obs	School	RaceTime ^①
1	CSULA	100.50
2	ASU	105.35
3	UoA	93.70
4	CSUF	100.25
5	UNLV	123.45
6	UCSD	86.47
7	UCLA	80.64
8	CPSLO	72.08
9	CPP	85.10
10	UHM	84.49

2.11 为每个观测读取多行原始数据

在典型的原始数据文件中，每行数据代表一个观测。但是，有的时候，一个观测的数据分散到了多行中。当 SAS 还没有为 INPUT 语句中所有变量读完数据就遇到了行结尾的时候，SAS 将自动地进入下一行继续读取。因此，你可以让 SAS 决定何时换行。但是如果你知道数据文件的多个原始数据行对应一个观测，则你最好显式告知 SAS 何时换行，而不是让其自行决定。用这种方法，你将不会在日志中遇到可疑的“SAS 进入了新的一行”



① 注意，时间以秒的方式打印出来。第 4.6 节将讨论如把这些时间格式化成分钟和秒。

的提示信息。如果你要告知 SAS 何时跳到下一行，你只需在 INPUT 语句中增加一个行指针。

行指针、斜线 (/) 和井号 n (#n)，就像一个指路牌一样告诉 SAS “这边走”。要为单个观测读取多行原始数据，你只需在 INPUT 语句中插入一个斜线 (/) 指示跳到下一个原始数据行。#n 行指针执行了相同的操作，除了你必须指定行号以外。#n 中的 n 代表了用于该观测的原始数据的行号，因此，#2 意味着为该观测进入原始数据的第二行。你甚至可以使用 #n 行指针返回到前面的行。例如，先读取第 4 行再读取第 3 行。斜线 (/) 比行指针更简单一些，但是 #n 行指针更灵活。

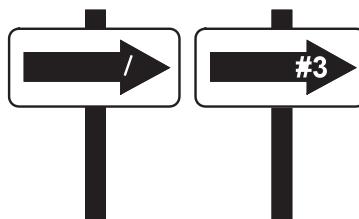


图 2-10

示例 一个同事正在规划他的下一个夏日假期，他想去温度适宜的地方。于是，他从一个气象数据库中获取了数据。不幸的是，他不知道如何正确地从数据库中导出数据，因此导出了一个非常奇怪的文件。

这个文件包含了阿拉斯加、佛罗里达和北卡罗来纳（如果你的同事决定访问最后一个地方，他可以顺便拜访一下 SAS 总部）7 月的气温信息。第一行包含了城市和州；第二行列出了正常情况下的最高温和最低温（华氏温度）；第三行包含了有史以来最高温和最低温：

```
Name AK  
55 44  
88 29  
Miami FL  
90 75  
97 65  
Raleigh NC  
88 68  
105 50
```

下面这段代码从名为 Temperature.dat 的文件中读取天气信息。

```
* 创建一个名字为highlow的SAS数据集;  
* 使用行指针, 读取数据文件;  
DATA highlow;  
    INFILE 'c:\MyRawData\Temperature.dat';  
    INPUT City $ State $  
        / NormalHigh NormalLow  
        #3 RecordHigh RecordLow;  
RUN;  
PROC PRINT DATA = highlow;  
    TITLE 'High and Low Temperatures for July';  
RUN;
```

INPUT 语句从第一行数据中读取 City 和 State 的值, 斜线 (/) 告知 SAS 在读取 NormalHigh 和 NormalLow 前移动到数据下一行的第 1 列。类似地, #3 告知 SAS 在读取 RecordHigh 和 RecordLow 之前移动到数据第三行的第 1 列。像以往一样, 这里有不止一种编写 INPUT 语句的方法。你可以用 #2 替代斜线 (/) 或用斜线 (/) 来替代 #3。

日志里出现了下面的提示

NOTE: 从 INFILE 'c:\MyRawData\Temperature.dat' 中读取了 9 条记录。

最小记录长度是 5。

最大记录长度是 10。

NOTE: 数据集 WORK.HIGHLOW 有 3 个观测和 6 个变量。

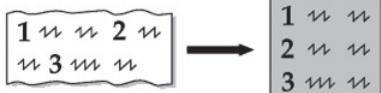
请注意, SAS 从输入文件读取了 9 行记录, 但 SAS 数据集里只有三条观测。通常情况下, 这会引起你的警觉, 但在里它证实了的确按计划为每个观测读取了三行数据。你应该始终检查 SAS 日志, 尤其当你使用行指针的时候。

表 2-8 是 PRINT 过程的输出结果, 你也可以使用 VIEWTABLE 查看该数据。

表 2-8 7 月温度记录表

Obs	City	State	NormalHigh	NormalLow	RecordHigh	RecordLow
1	Nome	AK	55	44	88	29
2	Miami	FL	90	75	97	65
3	Raleigh	NC	88	68	105	50

2.12 从每行原始数据读取多个观测



这应该是数据的“墨菲定律”：无论数据能够以何种形态存在，都会存在这种数据。正常情况下，SAS 假设每行原始数据代表的观测不会超过一个。如果每行原始数据有多个观测，你可以在 INPUT 语句结尾处使用双尾 @ 符号（@@）。这个行固定标识符就像一个停车标志一样告诉 SAS：“停下，留在原始数据的当前行。”SAS 将停留在那行数据，继续读取观测直到数据读取完毕或者遇到了没有双尾 @ 符号的 INPUT 语句为止。



图 2-11

示例 假设你的一位同事正在规划假期，他得到了一个包含他所想去的三个城市降雨量（英寸）的数据文件。该文件包含每个城市的名称、所在州名称、7月的平均降雨量以及7月有可测量降水的平均天数。该原始数据文件看起来如下所示：

```
Nome AK 2.5 15 Miami FL 6.75
18 Raleigh NC . 12
```

请注意，在该数据文件中，第一行在第二个观测的中间结束。下面的程序从名为 Precipitation.dat 的文件中读取这些数据。它使用了 @@，因此 SAS 不会为每个观测自动进入原始数据的新一行。

```
*从每个记录中输入多个观测;
DATA rainfall;
  INFILE 'c:\MyRawData\Precipitation.dat';
    INPUT City $ State $ NormalRain MeanDaysRain @@;
  RUN;
  PROC PRINT DATA = rainfall;
    TITLE 'Normal Total Precipitation and';
```

```
TITLE2 'Mean Days with Precipitation for July';
RUN;
```

下面的提示将会出现在日志中：

```
NOTE: 从 INFILE 'c:\MyRawData\Precipitation.dat' 中读取了 2 条记录。
      最小记录长度是 18。
      最大记录长度是 29。
```

```
NOTE: INPUT 语句到达一行的末尾, SAS 已转到新的一行。
```

```
NOTE: 数据集 WORK.RAINFALL 有 3 个观测和 4 个变量。
```

仅仅从原始数据文件中读取了两行记录，RAINFAL 数据集却包含三个观测。日志里也包含了一个提示，指明当 INPUT 语句到达行末尾的时候 SAS 进入了新的一行。这意味着 SAS 在一个观测的中间就走到了一行数据的末尾，它将继续读取下一行原始数据。正常情况下，这些信息说明这里有问题，但是在这个例子里，这正是你所需要的。

表 2-9 是 PRINT 过程的输出结果，你也可以用 VIEWTABLE 查看该数据。

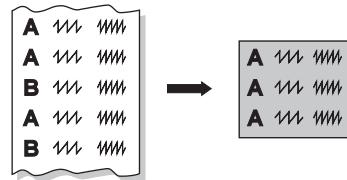
表 2-9 Normal Total Precipitation and Mean Days with Precipitation for July

Obs	City	State	NormalRain	MeanDaysRain
1	Nome	AK	2.50	15
2	Miami	FL	6.75	18
3	Raleigh	NC	.	12

2.13 读取原始数据文件的一部分

有的时候，你可能会发现你需要读取一个比较大的数据文件中的一小部分。例如，当你读取美国人口普查数据时，你可能只需要年薪在 \$225 000 以上、居住在 Walla Walla, Washington 的女性户主数据。当然，你也可以读取全部数据，然后再把不需要的丢掉，但是那将浪费你的时间。

幸运的是，你无须读取所有的数据，就可以告知 SAS 是否保留某条观测。取而代之的是，你可以只读取需要的变量以确定是否保留当前的观测，然后用一个 @ 符号（也被称作单尾 @）结束 INPUT 语句。该符号告知 SAS 保持那行原始数据。在单尾 @ 保持该



行数据时，你可以使用 IF 语句来测试该观测，以查看它是否是你要保留的观测。如果是，你可以使用第二条 INPUT 语句为其余变量读取数据。如果没有单尾 @，SAS 将自动地为每个 INPUT 语句读取下一行原始数据。

单尾 @ 和列指针 @n(在第 2.9 节中介绍)非常相似。通过在 @ 符号后面指定一个数值，你可以告知 SAS 移动到指定的列。如果 @ 后面没有指定列，那就好像你告知 SAS “敬请期待更多信息，请勿换台！” SAS 将保持住该行数据直到 DATA 步结束或者遇到一个没有单尾 @ 的 INPUT 语句为止。

示例 你想要读取一个包含本地高速公路和地面街道交通信息的文件的一部分。数据包含街道类型、街道名称、在早晨平均每小时通过的车辆数量以及在晚上平均每小时通过的车辆数量。这是原始数据文件：

freeway	408	3684	3459
surface	Martin Luther King Jr. Blvd.	1590	1234
surface	Broadway	1259	1290
surface	Rodeo Dr.	1890	2067
freeway	608	4583	3860
freeway	808	2386	2518
surface	Lake Shore Dr.	1590	1234
surface	Pennsylvania Ave.	1259	1290

假如你只是想看高速公路的数据，你可以使用下面这段程序来读取原始数据文件 Traffic.dat。

```
*使用单尾 @ , 然后删除地面街道;
DATA freeways;
  INFILE 'c:\MyRawData\Traffic.dat';
  INPUT Type $ @;
  IF Type = 'surface' THEN DELETE;
  INPUT Name $ 9-38 AMTraffic PMTraffic;
RUN;
PROC PRINT DATA = freeways;
  TITLE 'Traffic for Freeways';
RUN;
```

注意，这里有两条 INPUT 语句。第一个用来读取字符变量 Type，并以一个 @ 符号结尾。当使用 IF 语句测试的时候，单尾 @ 保持了每行数据。第二条 INPUT 语句读取了 Name (第 9 ~ 38 列)、AMTraffic 和 PMTraffic。如果一个观测的 Type 变量值为 surface，则第二条 INPUT 语句不会被执行。SAS 会回到 DATA 步的开始位置，然后处理下一个观测。它

不会将不需要的观测添加到 FREEWAYS 数据集中（立即执行，毫无余地）。

当你运行该程序的时候，日志里将包含下面两段提示。一个说明了从输入文件中读取了 8 行记录；另一个则说明了新的数据集只包含了 3 个观测。

NOTE: 从 INFILE 'c:\MyRawData\Traffic.dat' 中读取了 8 条记录。

最小记录长度是 47。

最大记录长度是 47。

NOTE: 数据集 WORK.FREeways 有 3 个观测和 4 个变量。

另外 5 个观测的 Type 变量值为 surface，IF 语句将它们删除了。下面是 PRINT 过程的输出结果，你也可以使用 VIEWTABLE 查看该数据。

表 2-10 Traffic for Freeways

Obs	Type	Name	AMTraffic	PMTraffic
1	freeway	408	3684	3459
2	freeway	608	4583	3860
3	freeway	808	2386	2518

对比单尾 @ 和双尾 @ 前一章节中介绍的双尾 @ 和单尾 @ 非常类似，它们都是行固定标识符。区别在于它们保持一行输入数据的时间长短不同。单尾 @ 为后续的 INPUT 语句保持一行数据，但是当 SAS 回到 DATA 步的顶部开始生成下一条观测时，它将释放那行数据。双尾 @ 即使在 SAS 开始生成新观测时，也为后续的 INPUT 语句保持那行数据。在这两种情况中，当 SAS 遇到了没有行固定标识符的后续 INPUT 语句时，该行数据都会被释放。

2.14 在 INFILE 语句中使用选项控制输入

到目前为止，我们在本章中已经看到了多种使用 INPUT 语句读取不同类型的原始数据的方法。当读取数据文件时，SAS 做了特定的假设。例如，SAS 从数据文件的第一行开始读取，如果 SAS 读完了一行数据，它将自动跳到下一行，继续为剩余的变量读取数据。在大多数情况下，这没有问题，但是有的数据文件无法基于这些默认的假设来读取。INFILE 语句中的选项可以改变 SAS 读取原始数据文件的方式。下面的选项对于读取特定类型的数据文件非常有用，将这些选项放在 INFILE 语句中文件名的后面。

FIRSTOBS= “FIRSTOBS=” 选项告知 SAS 从第几行开始读取数据。这对于那些在开头有描述性文本和头信息的数据文件来说非常有用，因为你需要跳过这些行从下面开始读取数据。例如，下面这个数据文件在前两行拥有描述信息。

```

Ice-cream sales data for the summer
Flavor      Location    Boxes sold
Chocolate   213         123
Vanilla     213         512
Chocolate   415         242

```

下面的程序使用 FIRSTOBS= 选项告知 SAS 从文件的第三行开始读取数据：

```

DATA icecream;
  INFILE 'c:\MyRawData\IceCreamSales.dat' FIRSTOBS = 3;
  INPUT Flavor $ 1-9 Location BoxesSold;
RUN;

```

OBS= 任何时候，当你想要读取部分数据文件时，你都可以使用 “OBS=” 选项。它告知 SAS 当到达数据文件中那一行时停止读取。注意，它不必和观测数一致。例如，如果你为每个观测读取两行原始数据，OBS=100 将会读取 100 行数据，而生成的 SAS 数据集只有 50 个观测。“OBS=” 选项可以和 “FIRSTOBS=” 一起使用以便从数据文件的中间读取数据。例如，假设以下冰激凌销售数据文件的结尾有一个不属于数据部分的评论。

```

Ice-cream sales data for the summer
Flavor      Location    Boxes sold
Chocolate   213         123
Vanilla     213         512
Chocolate   415         242
Data verified by Blake White

```

通过使用 FIRSTOBS=3 和 OBS=5，SAS 将从第 3 行数据开始读取，直到读完第 5 行数据就停下来。

```

DATA icecream;
  INFILE 'c:\MyRawData\IceCreamSales.dat' FIRSTOBS = 3 OBS=5;
  INPUT Flavor $ 1-9 Location BoxesSold;
RUN;

```

MISSOVER 默认情况下，当 SAS 读完一行数据后，如果 INPUT 语句中还有一些变量没有赋值，SAS 将会去下一数据行读取数据。“MISSOVER” 选项告知 SAS：当一

行数据读完的时候不要转到下一行，而是为其余的变量分配缺失值。该选项可能对下面这类文件比较有用。该文件包含了自定进度课程的测验分数。由于不是每个学生都完成了所有测验，所以有些学生比其他人拥有更多的分数。

```
Nguyen    89 76 91 82  
Ramos     67 72 80 76 86  
Robbins   76 65 79
```

下面这段代码读取了 5 个测验分数，为没有完成的测验分配了缺失值。

```
DATA class102;  
  INFILE 'c:\MyRawData\AllScores.dat' MISSOVER;  
  INPUT Name $ Test1 Test2 Test3 Test4 Test5;  
RUN;
```

TRUNCOVER 当你使用列输入或者格式化输入读取数据，并且一些数据行比其他行短的时候，你需要使用“TRUNCOVER”选项。如果一个变量字段超过了数据行的结尾，默认情况下，SAS 将转到下一行继续读取数据。该选项告知 SAS 为变量读取数据，直到遇到了数据行的结尾，或者遇到了在格式或列范围指定的最后一个列，二者以先遇到者为准。下面这个文件包含了地址信息，因为地址名称有内嵌的空格，所以必须使用列输入或者格式化输入。请注意，以下数据行的长度均不相同。

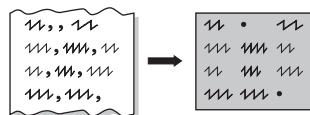
```
John Garcia  114 Maple Ave.  
Sylvia Chung 1302 Washington Drive  
Martha Newton 45 S.E. 14th St.
```

下面的程序使用列输入读取地址文件。因为有些地址在变量 Street 的作用域（第 22 ~ 37 列）结束之前就停止了，你需要使用“TRUNCOVER”选项。如果没有“TRUNCOVER”选项，SAS 将在第一条和第三条记录尝试转到下一行读取 Street。

```
DATA homeaddress;  
  INFILE 'c:\MyRawData\Address.dat' TRUNCOVER;  
  INPUT Name $ 1-15 Number 16-19 Street $ 22-37;  
RUN;
```

TRUNCOVER 和 MISSOVER 类似。如果数据行在变量作用域开始之前就结束了，它们都会为变量分配缺失值。但是，如果数据行在变量作用域中间结束时，TRUNCOVER 将尽量读取可用数据，而 MISSOVER 则直接为变量分配一个缺失值。

2.15 使用 DATA 步读取分隔文件



分隔文件是一种使用特殊字符分隔数据值的原始数据文件。许多程序都能把数据保存成分隔文件，通常使用逗号或者制表符作为分隔符。SAS 为 INFILE 提供两个使其更易读取分隔数据文件的选项：“DLM=” 选项和 “DSD” 选项。

“DLM=” 选项 如果你使用列表输入读取数据，DATA 步期望你的文件用空格分隔数据。INFILE 语句中的 “DELIMITER=” 或者 “DLM=” 选项使你能够读取使用其他分隔符的文件。逗号和制表符是数据文件中常见的分隔符，但你也可以读取使用任何分隔符的数据文件，只需把分隔符放在 “DLM=” 选项后面的引号中（例如 DLM='&'）。如果你的分隔符是一个字符串，请使用 “DLMSTR=” 选项。

示例 下面的文件是使用逗号分隔的，学生名称后面跟的是暑期阅读计划中每周的读书数量。

```
Grace,3,1,5,2,6
Martin,1,2,4,1,3
Scott,9,10,4,8,6
```

下面的程序使用列表输入读取该书籍数据文件，并指定分隔符为逗号。

```
DATA reading;
  INFILE 'c:\MyRawData\Books.dat' DLM = ',';
  INPUT Name $ Week1 Week2 Week3 Week4 Week5;
RUN;
```

如果还是相同的数据，但是使用制表符代替逗号作为分隔符，那么你可以使用下面的程序读取这个文件。该程序使用了 DLM='09'X 选项。在 ASCII 码中，09 是制表符的十六进制形式。符号 ‘09’ X 代表了 09 的十六进制值。如果你的计算机使用的是 EBCDIC (IBM 大型机) 而不是 ASCII，那么请使用 DLM='05'X。

```
DATA reading;
  INFILE 'c:\MyRawData\Books.txt' DLM = '09'X;
  INPUT Name $ Week1 Week2 Week3 Week4 Week5;
RUN;
```

默认情况下，SAS 将两个或者更多的连续分隔符解释为一个分隔符。如果你的文件有缺失值，并且连续的两个分隔符代表缺失值，那么你将需要用到 INFILE 语句中的“DSD”

选项。

“DSD”选项 INFILE 语句中的“DSD”（分隔符敏感数据）选项为你做三件事情。首先，它忽略用引号括起来的数据值中的分隔符。其次，它不会把引号作为数值的一部分读取。最后，它把两个连续的分隔符视为缺失值。“DSD”选项假设你的分隔符是逗号。如果分隔符不是逗号，你可以配合“DSD”选项使用“DLM=”选项指定其他的分隔符。例如，为了读取一个制表符分隔的 ASCII 文件，并且文件中使用连续的两个制表符表示缺失值，你可以使用下面语句：

```
INFILE 'file-specification' DLM = '09'X DSD;
```

CSV 文件 逗号分隔值文件（也称为 CSV 文件）是可以使用“DSD”选项读取的常见文件类型。许多程序（如 Microsoft Excel）可以将数据保存成 CSV 格式。CSV 文件使用逗号作为分隔符，连续的逗号表示缺失值。如果数据值包含逗号，则数据值会被放在引号中。

示例 下面这个示例展示了如何使用“DSD”选项读取一个 CSV 文件。Jerry 的咖啡馆雇用了本地的乐队来吸引顾客。乐队每天晚上在咖啡馆里演出，Jerry 分别记录了他们吸引顾客的数量。下面的数据表中包含了乐队的名称、演出日期以及分别在 8 p.m.、9 p.m.、10 p.m. 和 11 p.m. 的在场顾客数量。

```
Lupine Lights,12/3/2012,45,63,70,  
Awesome Octaves,12/15/2012,17,28,44,12  
"Stop, Drop, and Rock-N-Roll",1/5/2013,34,62,77,91  
The Silveyville Jazz Quartet,1/18/2013,38,30,42,43  
Catalina Converts,1/31/2013,56,,65,34
```

请注意，有一个乐队的名称包含逗号，该乐队名称被放在引号中。最后一个乐队缺失 9 p.m. 时候的值，连续的两个逗号表示该缺失值。在 INFILE 语句中使用“DSD”选项可以读取该数据文件。这里还需要谨慎一点，当使用“DSD”选项的时候，如果在数据行的末尾有可能出现缺失值（就像上面数据文件的第一行一样），需要一并使用“MISSOVER”选项。“MISSOVER”选项告知 SAS 当一行数据不够的时候，不要跳到下一行继续读取。下面就是读取该数据文件的程序：

```
DATA music;  
  INFILE 'c:\MyRawData\Bands.csv' DLM = ',' DSD MISSOVER;  
  INPUT BandName :$30. GigDate :MMDDYY10. EightPM NinePM TenPM ElevenPM;  
  RUN;
```

```
PROC PRINT DATA = music;
  TITLE 'Customers at Each Gig';
  RUN;
```

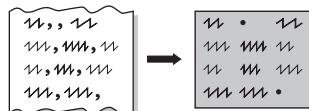
注意，对于 BandName 和 GigDate 我们使用了带有冒号修饰的输入格式。冒号修饰符告知 SAS 读取输入格式的长度（BandName 为 30 个字符，GigDate 为 10 个字符）或者直到遇到分隔符为止，二者以先遇到者为准。因为乐队名称的长度大于默认的 8 个字符，我们为 BandName 使用：\$ 30. 输入格式以读取最多 30 个字符。

表 2-11 是 PRINT 过程的输出结果，你也可以在 VIEWTABLE 窗口中查看该数据。

表 2-11 Customers at Each Gig

Obs	BandName	GigDate ^①	EightPM	NinePM	TenPM	ElevenPM
1	Lupine Lights	19330	45	63	70	.
2	Awesome Octaves	19342	17	28	44	12
3	Stop, Drop, and Rock-N-Roll	19363	34	62	77	91
4	The Silveyville Jazz Quartet	19376	38	30	42	43
5	Catalina Converts	19389	56	.	65	34

2.16 使用 IMPORT 过程读取分隔文件



我们猜想到目前为止你已经意识到了在 SAS 中通常有不止一种方法来达成相同的结果。在前面的章节中，我们向你展示了如何使用 DATA 步读取分隔文件。现在，我们将向你展示如何使用另外一种不同的方法读取分隔文件：使用 IMPORT 过程。IMPORT 过程在 UNIX 和 Windows 操作环境上都可以使用。

PROC IMPORT 为你做了一些事情来帮助你更容易地读取特定类型的数据文件。PROC IMPORT 扫描你的数据文件（默认扫描前 20 行），然后自动确定变量的类型（字符或者数值）。它还能为字符类型指定长度以及识别一些日期格式。PROC IMPORT 将会把数据文件中两个连续的分隔符视为缺失值，读取引号中包含的数据值，以及当数据读完的时候为其余的变量分配缺失值。而且，如果你需要，你可以使用数据文件中的第

① 注意，这些日期被打印成了从 1960 年 1 月 1 日开始的天数。第 4.6 节将讨论如何把它们格式化可读的日期。

一行作为变量的名称。

IMPORT 过程的一般形式如下：

```
PROC IMPORT DATAFILE = 'filename' OUT = data-set;
```

filename 是你想要读取的文件的名称，*data-set* 是你要创建的 SAS 数据集的名称。

SAS 将通过文件的扩展名来确定文件的类型，如下表所示：

文件类型	扩展名	DBMS 标识符
逗号分隔的	.csv	CSV
制表符分隔的	.txt	TAB
使用逗号和制表符以外的分隔符的		DLM

如果你的文件没有适当的扩展名，或者你的文件是某种 DLM 类型，你必须在 PROC IMPORT 中使用 “DBMS=” 选项。如果已经有了一个和 “OUT=” 选项中指定的名称同名的 SAS 数据集，并且你想要覆盖它，你需要使用 “REPLACE” 选项。这是在 PROC IMPORT 中使用 “REPLACE” 选项和 “DBMS” 选项的一般形式：

```
PROC IMPORT DATAFILE = 'filename' OUT = data-set
    DBMS = identifier REPLACE;
```

可选的语句 一些文件类型需要用一些额外的指令才能正确读取。如果数据文件不是在第一行开始，可以使用 DATAROWS 语句。如果分隔符不是逗号、制表符或者空格，可以使用 DELIMITER 语句。如果你的文件只包含数据，没有标题行，请使用 “GETNAMES=NO” 语句分配默认的变量名称。如果你的数据在前 20 行全是缺失值或者没有代表性的数据，你可能需要 GUESSINGROWS 语句以确保变量被分配了正确的数据类型和长度。

DATAROWS = <i>n</i> ;	从第 <i>n</i> 行开始读取数据，默认是 1。
DELIMITER = ' <i>delimiter-character</i> ';	DLM 文件的分隔符，默认是空格。
GETNAMES = NO;	不要从输入文件的第一行获取变量名称，默认是 YES。 如果是 NO，变量名称为 VAR1、VAR2、VAR3 等。
GUESSINGROWS = <i>n</i> ;	使用 <i>n</i> 行来确定变量的类型，默认为 20。

下面展示了包含 “GETNAMES=NO” 语句的 PROC IMPORT 一般形式：

```
PROC IMPORT DATAFILE = 'filename' OUT = data-set;
    GETNAMES = NO;
```

示例 下面的示例使用了 Jerry 咖啡馆的数据，Jerry 雇用了本地的乐队来吸引顾客。Jerry 为每个乐队记录了整晚在场顾客的数量。数据包含乐队的名称、演出日期以及分别在 8 p.m、9 p.m、10 p.m 和 11 p.m 的在场顾客数量。注意，其中的一个乐队 “Stop, Drop, and Rock-N-Roll” 名称中间包含了逗号。当一个数值中有分隔符时，它必须被放在引号中。

```
Band Name,Gig Date,Eight PM,Nine PM,Ten PM,Eleven PM
Lupine Lights,12/3/2012,45,63,70,
Awesome Octaves,12/15/2012,17,28,44,12
"Stop, Drop, and Rock-N-Roll",1/5/2013,34,62,77,91
The Silveyville Jazz Quartet,1/18/2013,38,30,42,43
Catalina Converts,1/31/2013,56,,65,34
```

下面这段程序读取了该数据文件，并在导入后打印该 SAS 数据集：

```
PROC IMPORT DATAFILE ='c:\MyRawData\Bands2.csv' OUT = music REPLACE;
RUN;
PROC PRINT DATA = music;
  TITLE 'Customers at Each Gig';
RUN;
```

表 2-12 是 PROC PRINT 的输出结果。注意，演出日期是一个易读的日期，这是因为 IMPORT 自动地为某些日期形式分配了输入格式和输出格式（请参见第 4.6 节中关于输出格式的介绍）。还要注意 PROC IMPORT 使用了第一行数据作为变量的名称。为了符合标准的 SAS 变量命名规则，名称中的空格被下划线所替代。你也可以使用 VIEWTABLE 查看该数据。

表 2-12 Customers at Each Gig

Obs	Band_Name	Gig_Date	Eight_PM	Nine_PM	Ten_PM	Eleven_PM
1	Lupine Lights	12/03/2012	45	63	70	.
2	Awesome Octaves	12/15/2012	17	28	44	12
3	Stop, Drop, and Rock-N-Roll	01/05/2013	34	62	77	91
4	The Silveyville Jazz Quartet	01/18/2013	38	30	42	43
5	Catalina Converts	01/31/2013	56	.	65	34

2.17 使用 IMPORT 过程读取 Excel 文件

如果你有 SAS/ACCESS Interface to PC Files，你可以使用 IMPORT 过程在 Windows 和 UNIX 操作环境中导入 Microsoft Excel 文件。在 Windows 操作环境中，有一种无须 SAS/ACCESS 就能读取 Microsoft Excel 文件的替代方法，那就是使用下一节中介绍的动态数据交换（DDE）。

下面是使用 IMPORT 过程读取 EXCEL 文件的一般形式：

```
PROC IMPORT DATAFILE = 'filename' OUT = data-set  
    DBMS = identifier REPLACE;
```

其中，filename 是你要读取的文件，data-set 是你要创建的 SAS 数据集的名称。“REPLACE”选项告知 SAS 替换“OUT=”选项中指定的 SAS 数据集（若存在的话）。“DBMS=”选项告知 SAS 要读取的 EXCEL 文件类型，它不是必需的。

DBMS 标识符 这里有一些 DBMS 标识符可用于读取 EXCEL 文件。三种最常用的标识符是 EXCEL、XLS 和 XLSX。在 UNIX 操作环境中，使用 XLS 标识符读取老版本的文件（扩展名为.xls），使用 XLSX 标识符读取新版本的文件（扩展名为.xlsx）。在 Windows 操作环境中，除了 XLS 和 XLSX 标识符以外，你还可以使用 EXCEL 标识符读取所有类型的 Excel 文件。与 XLS 和 XLSX 标识符相比，EXCEL 使用了不同的读取技术，所以结果可能会有所不同。默认情况下，XLS 和 XLSX 标识符，相较于 EXCEL 标识符，会查看更多的数据行来确定列的类型。如果你的 Windows 计算机同时有 64 位和 32 位应用程序，那么不是所有的标识符都适用于你。此外，某些计算机配置可能需要预先安装 PC 文件服务器。PC 文件服务器使用 EXCELCS 标识符。详细信息，请参见 SAS 帮助文档。

可选的语句 如果你的文件中有多个工作表，你可以使用下面语句指定读取哪个工作表。

```
SHEET = "sheet-name";
```

如果你只想读取工作表中特定的单元格，你可以指定一个区域。该区域可以是命名区域（如果已经定义），或者你也可按以下方式指定所在区域的左上和右下单元格。

```
RANGE = "sheet-name$UL:LR";
```

默认情况下，IMPORT 过程将使用电子表格的第一行作为变量的名称。如果你不想这样做，你可以向该过程添加以下语句（仅限 EXCEL 标识符）。SAS 会将这些变量命名为 F1、F2 等。

```
GETNAMES = NO;
```

当使用 Excel 标识符的时候，如果有一列同时包含数值和字符值，默认情况下，数值将被转换成缺失值。为了将数值读取成字符类型（而不是缺失值），可以使用下面的语句。

```
MIXED = YES;
```

示例 假设你有如表 2-13 所示的 Microsoft Excel 电子表格，它包含了本地小联盟棒球队比赛中的洋葱圈销售数量。数据表包含客队名称、小卖部的销售数量、露天看台的销售数量以及每个队的击打和奔跑数量。

表 2-13 本地小联盟棒球比赛中洋葱圈销售数量

	A	B	C	D	E	F	G
1	VisitingTeam	C Sales	B Sales	Our Hits	Their Hits	Our Runs	Their Runs
2	Columbia Peaches	35	67	1	10	2	1
3	Plains Peanuts	210		2	5	0	2
4	Gilroy Garlics	15	1035	12	11	7	6
5	Sacramento Tomatoes	124	85	15	4	9	1
6							
7							

下面这段程序使用 IMPORT 过程和 XLS DBMS 标识符读取了该 Microsoft Excel 文件。

```
*使用PROC IMPORT读取Excel电子表格;
PROC IMPORT DATAFILE = 'c:\MyExcel\OnionRing.xls' DBMS=XLS OUT =
sales;
RUN;
PROC PRINT DATA = sales;
  TITLE 'SAS Data Set Read From Excel File';
RUN;
```

下面是 PROC PRINT 的输出结果。请注意，变量名称是如何从电子表格的第一行获取的。空格被替换成了下划线以符合标准的 SAS 变量命名规范（不多于 32 个字符，以字母或者下划线开头，只包含字母、下划线和数字）。你也可以使用 VIEWTABLE 查看该数据。

表 2-14 读取自 Excel 表格的 SAS 数据集

Obs	Visiting_Team	C_Sales	B_Sales	Our_Hits	Their_hits	Our_Runs	Their_Runs
1	Columbia Peaches	35	67	1	10	2	1
2	Plains Peanuts	210	.	2	5	0	2
3	Gilroy Garlics	15	1035	12	11	7	6
4	Sacramento Tomatoes	124	85	15	4	9	1

2.18 临时和永久 SAS 数据集

SAS 数据集以两种方式存在：临时的和永久的。临时 SAS 数据集仅在当前的作业或者会话中存在，当前作业或者会话结束时，它会被自动地清除掉。如果 SAS 数据集是永久的，并不是说它会永远存在，而是在你的作业或者会话结束后，它会被保留下来。

每种类型的数据集都有它自己的优势。有时候你希望将数据集保存下来供以后使用，有时候你则不希望这么做。在这本书中，大多数例子使用了临时数据集，因为我们不想使你的磁盘变得杂乱无章。但是，一般情况下，如果你希望多次使用同一个数据集，相比较于每次都创建一个临时数据集来说，更高效的做法是将其保存成永久数据集。

SAS 数据集名称 所有的 SAS 数据集都有一个两级的名称，例如 WORK.BIKESALES，两个层级之间被一个句点分隔。SAS 数据集名称的第一个层级（本例中为 WORK）被称为逻辑库引用名（libref，SAS data library reference 的缩写形式）。逻辑库引用名就像指向特定位置的箭头。有时候，逻辑库引用名指向一个物理位置，例如闪存盘或者 CD，有的时候它指向一个逻辑位置，例如目录或者文件夹。SAS 数据集名称中的第二个层级（本例中为 BIKESALES）是在逻辑库中唯一标识该数据集的成员名称。

逻辑库引用名和成员名称都遵循有效 SAS 名称的标准规则。它们必须以字母或者下划线开始，且只包含字母、数字或者下划线。然而，逻辑库引用名的长度不能超过 8 个字符，成员名称却可以最长达 32 个字符。

你从不显式告知 SAS 创建临时数据集还是永久数据集。当创建数据集的时候，你通过数据集的名称来暗示系统要创建的类型。大多数数据集都是在 DATA 步中创建的，但是 PROC 步也可以创建数据集。如果你指定了两级名称（并且逻辑库引用名不为 WORK），则你的数据集是永久的。如果你仅指定了一级数据集名称（就像本书中大多

数例子一样），则你的数据集是临时的。SAS 将使用你的一级名称作为成员名称，然后自动地附上逻辑库引用名 WORK。按照定义，任何逻辑库引用名为 WORK 的 SAS 数据集都是临时的，在作业或者会话结束时，将被 SAS 清除掉。下面是一些 DATA 语句示例以及它们所创建数据集的特性。

DATA 语句	逻辑库引用名	成员名称	类型
DATA ironman;	WORK	ironman	临时
DATA WORK.tourdefrance;	WORK	tourdefrance	临时
DATA Bikes.doublecentury;	Bikes	doublecentury	永久

临时 SAS 数据集 下面这段程序创建并打印了一个名为 DISTANCE 的临时 SAS 数据集：

```
DATA distance;
  Miles = 26.22;
  Kilometers = 1.61 * Miles;
RUN;
PROC PRINT DATA = distance;
RUN;
```

请注意，逻辑库引用名 WORK 并没有出现在 DATA 语句中。因为数据集使用了一级名称，所以 SAS 分配了默认的逻辑库 WORK，且使用 DISTANCE 作为该逻辑库中的成员名称。下面这段日志中的注释提示了完整的两级名称。

NOTE: 数据集 WORK.DISTANCE 有 1 个观测和 2 个变量。

永久 SAS 数据集 在你使用逻辑库引用名之前，你需要先定义它。你可以使用“新建逻辑库”窗口定义逻辑库，也可以使用 LIBNAME 语句（在下一节中详细介绍）定义逻辑库或者使用直接引用（在第 2.20 节中介绍）让 SAS 为你定义逻辑库引用名^①。

要打开“新建逻辑库”窗口，从菜单栏中选择工具 ▶ 新建逻辑库。在图 2-12 所示的“新建逻辑库”窗口中定义的 BIKES 逻辑库指向了 C 盘（Windows）下“MySASLib”文件夹下的“Ruiz Racing Bicycles”文件夹。

^① 当在 z/OS 环境中使用批处理时，你可能还会用到 Job Control Language (JCL)，这个 DDname 就是你的逻辑库引用名。



图 2-12

使用“新建逻辑库”窗口定义的逻辑库引用名将会出现在左图所示的 SAS 资源管理器的“当前逻辑库”窗口中，如图 2-13 所示。如果你双击该逻辑库图标，该逻辑库中的所有 SAS 数据集将被列在该逻辑库的“内容”窗口中。



图 2-13

下面这段程序，除了是创建一个永久 SAS 数据集以外，其余和前一个程序是相同的。请注意，两级名称出现在 DATA 语句和“DATA=” 选项中。

```

DATA Bikes.distance;
  Miles = 26.22;
  Kilometers = 1.61 * Miles;
RUN;
PROC PRINT DATA = Bikes.distance;
RUN;

```

这次，日志里包含了下面这条提示：

NOTE: 数据集 BIKES.DISTANCE 有 1 个观测和 2 个变量。

这是一个永久 SAS 数据集，因为逻辑库引用名不是 WORK。

2.19 通过 LIBNAME 语句使用永久 SAS 数据集



逻辑库引用名相当于是对 SAS 数据逻辑库位置的昵称。当你使用逻辑库引用名作为 SAS 数据集的第一级名称时，SAS 就会知道到那个位置去寻找 SAS 数据集。本节将向你展示如何使用 LIBNAME 语句定义逻辑库引用名，这是 SAS 里最普遍使用的（因此也是最便捷的）创建逻辑库引用名的方法。你也可以使用“新建逻辑库”窗口（在前面一节中介绍过）或者操作环境控制语言（在某些计算机上适用）^① 来定义逻辑库引用名。LIBNAME 语句的基本形式为：

```
LIBNAME libref 'your-SAS-data-library';
```

在 LIBNAME 关键字之后，先指定逻辑库引用名，然后是包含在引号中的永久 SAS 数据集的位置。逻辑库引用名必须不能超过 8 个字符，以字母或者下划线开头，且只包含字母、数字或者下划线。以下是各个操作环境中 LIBNAME 语句的一般形式：

Windows:

```
LIBNAME libref 'drive:\directory';
```

UNIX:

```
LIBNAME libref '/home/path';
```

z/OS:

```
LIBNAME libref 'data-set-name';
```

^① 当在 z/OS 环境中使用批处理时，你可能还会用到 Job Control Language (JCL)，这个 DDname 就是你的逻辑库引用名。

创建永久 SAS 数据集 下面这个示例创建了一个永久 SAS 数据集，它包含了木兰科树木的信息。对于每种树，原始数据文件包含了学名、常用名、最大的高度、使用种子种植时第一次开花的年龄、是常绿植物还是落叶植物以及花朵的颜色。

```
M. grandiflora    Southern Magnolia  80  15  E  white
M. campbellii          80  20  D  rose
M. liliiflora    Lily Magnolia     12   4  D  purple
M. soulangiana   Saucer Magnolia  25   3  D  pink
M. stellata      Star Magnolia    10   3  D  white
```

下面这段程序创建了一个逻辑库引用名 PLANTS，它指向了 C 盘（Windows）的 MySASLib 目录。然后，它从名为 Mag.dat 的文件中读取原始数据，并创建了一个名为 MAGNOLIA 的永久数据集，该数据集存储在 PLANTS 逻辑库中。

```
LIBNAME plants 'c:\MySASLib';
DATA plants.magnolia;
  INFILE 'c:\MyRawData\Mag.dat';
  INPUT ScientificName $ 1-14 CommonName $ 16-32 MaximumHeight
        AgeBloom Type $ Color $;
RUN;
```

日志中包含了两级数据集名称的提示。

```
NOTE: 数据集 PLANTS.MAGNOLIA 有 5 个观测和 6 个变量。
```

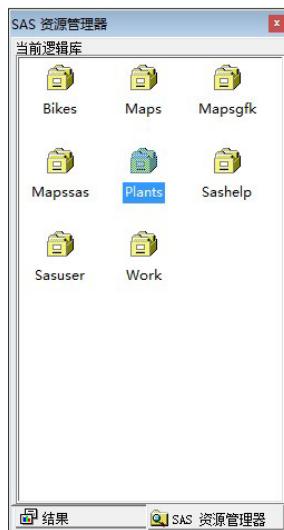


图 2-14

使用 LIBNAME 语句定义的逻辑库引用名将出现在 SAS 资源管理器的“当前逻辑库”窗口中。如果你双击该逻辑库图标，该逻辑库中所有的 SAS 数据集将被列在该逻辑库的“内容”窗口中。

如果你将计算机上的文件目录打印出来，你不会看到名为 PLANTS . MAGNOLIA 的文件。那是因为操作环境都有它们自己的文件命名系统。当运行在 Windows 或者 UNIX 环境中，该数据集的文件名是 magnolia.sas7bdat。在 z/OS 环境中，文件名是 LIBNAME 语句所指定的数据集名称。

读取永久 SAS 数据集 要使用永久 SAS 数据集，可以在你的程序中包含一条 LIBNAME 语句，并通过其两级名称引用它。例如，如果你要回去打印上一个例子中创建的永久数据集，你可以使用下面的语句：

```
LIBNAME example 'c:\MySASLib';
PROC PRINT DATA = example.magnolia;
  TITLE 'Magnolias';
RUN;
```

这次 LIBNAME 语句中的逻辑库引用名是 EXAMPLE，而不是上个例子中的 PLANTS，但是它指向了和以前一样的位置，即 C 盘上的 MySASLib 目录。逻辑库引用名可以更改，但是成员名称（MAGNOLIA）必须相同。

输出结果如表 2-15 所示。

表 2-15 木兰科树木表

Obs	ScientificName	CommonName	MaximumHeight	AgeBloom	Type	Color
1	M. grandiflora	Southern Magnolia	80	15	E	white
2	M. campbellii		80	20	D	rose
3	M. liliiflora	Lily Magnolia	12	4	D	purple
4	M. soulangiana	Saucer Magnolia	25	3	D	pink
5	M. stellata	Star Magnolia	10	3	D	white

2.20 通过直接引用使用永久 SAS 数据集

如果你不想为建立逻辑库引用名和定义 SAS 逻辑库所烦恼，但是你仍然想要使用永

久 SAS 数据集，那么你可以使用直接引用。直接引用仍是使用 SAS 逻辑库，但是无须你自己定义逻辑库，SAS 可以帮你做这件事。

直接引用很容易使用。只需取得该文件在你的操作环境中的名称，将它包含在引号中并放在你的程序中即可。引号告知 SAS 那是一个永久 SAS 数据集。下面是在不同的操作环境中，使用 DATA 语句创建永久 SAS 数据集的一般形式：

```
Windows:           DATA 'drive:\directory\filename';
UNIX:             DATA '/home/path/filename';
z/OS:             DATA 'data-set-name';
```

对于基于目录的操作环境而言，如果你省略了目录或者路径，SAS 将使用当前的工作目录。例如，下面这个语句将在你的当前工作目录中创建一个名为 TREES 的永久 SAS 数据集。

```
DATA 'trees';
```

对于 UNIX 操作环境，默认情况下，你的当前目录是你启动 SAS 的目录。你可以通过在菜单栏中选择 **工具** ▶ **选项** ▶ **更改当前文件夹** 来更改 SAS 会话的当前目录。在 Windows 中，当前工作目录的名称显示在 SAS 窗口的底部。你可以更改当前 SAS 会话的目录，方法是通过双击该目录的名称以打开“更改文件夹”窗口。

示例 下面的例子创建了一个包含木兰科树木信息的永久 SAS 数据集。对于每种树，原始数据文件包含了学名、常用名、最大的高度、使用种子种植时第一次开花的年龄、是常绿植物还是落叶植物以及花朵的颜色。

```
M. grandiflora Southern Magnolia 80 15 E white
M. campbellii                 80 20 D rose
M. liliiflora Lily Magnolia    12   4 D purple
M. soulangiana Saucer Magnolia 25   3 D pink
M. stellata Star Magnolia     10   3 D white
```

下面这段程序从名为 Mag.dat 的文件中读取原始数据，并创建一个名为 MAGNOLIA 的永久 SAS 数据集。该数据集存储在 C 盘（Windows）的 MySASLib 目录中。

```
DATA 'c:\MySASLib\magnolia';
INFILE 'c:\MyRawData\Mag.dat';
INPUT ScientificName $ 1-14 CommonName $ 16-32 MaximumHeight
      AgeBloom Type $ Color $;
RUN;
```

如果查看 SAS 日志，你将会看到下面的提示：

NOTE: 数据集 c:\MySASLib\magnolia 有 5 个观测和 6 个变量。

这是一个永久 SAS 数据集，因此 SAS 不会清除它。如果你列出 MySASLib 目录中的文件，你将会看到一个名为 magnolia.sas7bdat 的文件。请注意，SAS 自动地附上了一个文件扩展名，即使该扩展名在 SAS 程序中不显示。

当你将数据集名称放在引号中时，你就是在使用直接引用，SAS 会创建一个永久 SAS 数据集。由于你还没有指定逻辑库引用名，SAS 为你创造了一个逻辑库引用名。你不需要知道 SAS 创造的这个逻辑库引用名，但是它确实存在，你可以在“当前逻辑库”窗口中看到它。运行前面的程序后，“当前逻辑库”窗口的内容如图 2-15 所示。SAS 创建了一个名为 WC000001 的逻辑库，它包含 MAGNOLIA 数据集。



图 2-15

使用直接引用读取 SAS 数据集 要使用直接引用读取永久 SAS 数据集，只需在要使用该 SAS 数据集的地方将该数据集的路径和名称包含在引号中即可。例如，为了打印 MAGNOLIA 数据集，你可以使用下面的语句：

```
PROC PRINT DATA = 'c:\MySASLib\magnolia';
  TITLE 'Magnolias';
RUN;
```

输出结果如表 2-16 所示。

表 2-16 木兰科树木表

Obs	ScientificName	CommonName	MaximumHeight	AgeBloom	Type	Color
1	M. grandiflora	Southern Magnolia	80	15	E	white
2	M. campbellii		80	20	D	rose
3	M. liliiflora	Lily Magnolia	12	4	D	purple
4	M. soulangiana	Saucer Magnolia	25	3	D	pink
5	M. stellata	Star Magnolia	10	3	D	white

2.21 列出 SAS 数据集中的内容

为了使用 SAS 数据集，你所需要做的就是告知 SAS 要使用的数据集的名称和位置，SAS 将帮你查看其中的内容。SAS 能做到这一点是因为 SAS 数据集是可以自我说明的。换句话说，SAS 自动地连同数据一起存储了有关该数据集的信息（也被称作描述符部分）。你无法使用文字处理程序在计算机屏幕上显示 SAS 数据集。但是，有一个简便的方法可以获取 SAS 数据集的描述，运行 CONTENTS 过程即可。

PROC CONTENTS 是一个简单的过程，你只需要输入关键字 PROC CONTENTS 并使用“DATA=” 选项指定数据集即可。

```
PROC CONTENTS DATA = data-set;
```

示例 下面的 DATA 步创建了一个数据集，所以我们可以运行 PROC CONTENTS：

```
DATA funnies (LABEL = 'Comics Character Data');
  INPUT Id Name $ Height Weight DoB MMDDYY8. @@;
  LABEL Id = 'Identification no.'
    Height = 'Height in inches'
    Weight = 'Weight in pounds'
    DoB = 'Date of birth';
  INFORMAT DoB MMDDYY8.;
  FORMAT DoB WORDDATE18.;
  DATALINES;
53 Susie   42 41 07-11-81 54 Charlie 46 55 10-26-54
55 Calvin 40 35 01-10-81 56 Lucy    46 52 01-13-55
;
```

```
*使用PROC CONTENTS描述数据集funnies;
PROC CONTENTS DATA = funnies;
RUN;
```

请注意，上面的 DATA 步在 DATA 语句中包含了一个“LABEL= 数据集”选项^①，以及一条 LABEL 语句。“LABEL= 数据集”选项给整个数据集设置了一个标签，而 LABEL 语句则为每个变量分配标签。这些可选的标签，最长不超过 256 个字符，使你能够为你的数据记录下比变量名称和数据集名称更多的详细信息。对于变量，如果在 DATA 步指定了 LABEL 语句，这段描述将被存储到数据集中，并且可以使用 PROC CONTENTS 打印出来。你也可在 PROC 步中使用 LABEL 语句定制你的报表，但是那些标签只在使用 PROC 步的过程中适用，并不会存储到数据集中。

该程序还包含了 INFORMAT 和 FORMAT 语句。你可以使用这些可选的语句来为变量关联输入格式和输出格式。正如同输入格式向 SAS 发出读取变量的特殊指令一样，输出格式向 SAS 发出写变量的特殊指令。如果在 DATA 步中指定了一条 INFORMAT 或 FORMAT 语句，则该输入格式或者输出格式的名称将存储到数据集中，并且可以使用 PROC CONTENTS 打印出来。FORMAT 语句如同 LABEL 语句一样，可以用在 PROC 步中定制你的报表，但是该输出格式的名称不会存储到数据集中^②。

PROC CONTENTS 的输出类似于数据集的目录（见表 2-17）。

表 2-17 CONTENTS PROCEDURE

❶ 数据集名	WORK.FUNNIES	❷ 观测	4
成员类型	DATA	❸ 变量	5
引擎	V9	索引	0
❹ 创建时间	Monday, March 19, 2012 09:23:43 PM	观测长度	40
上次修改时间	Monday, March 19, 2012 09:23:43 PM	删除的观测	0
保护		已压缩	NO
数据集类型		已排序	NO
❺ 标签	Comics Character Data		
数据表示法 Representation	WINDOWS_32		
编码	wlatin1 Western (Windows)		

① 更多关于数据集选项的信息，请参见第 6.11 节。

② 第 4.6 节和第 4.7 节将更彻底地讨论标准 SAS 格式。

(续表)

引擎 / 主机相关的信息	
数据集页面大小	4096
数据集页数	1
首数据页	1
每页最大观测数	101
首数据页的观测数	4
数据集修复数	0
文件名	C:\Users\Lora\AppData\Local\Temp\SAS Temporary Files\funnies.sas7bdat
创建版本	9.0401M4
创建主机	W32_VSPRO

按字母排序的变量和属性列表						
#	变量	⑥ 类型	⑦ 长度	⑧ 输出格式	⑨ 输入格式	⑩ 标签
5	DoB	Num	8	WORDDATE18.	MMDDYY8.	Date of birth
3	Height	Num	8			Height in inches
1	Id	Num	8			Identification no.
2	Name	Char	8			
4	Weight	Num	8			Weight in pounds

输出内容先是有关你的数据集的信息，然后描述了每个变量。

关于该数据集

- ① 数据集名称
- ② 观测数
- ③ 变量数
- ④ 创建日期
- ⑤ 数据集标签（如果有的话）

关于每个变量

- ⑥ 类型（数值或者字符）
- ⑦ 长度（以字节为单位的存储大小）
- ⑧ 用于打印的输出格式（如果有的话）
- ⑨ 用于输入的输入格式（如果有的话）
- ⑩ 变量标签（如果有的话）

3

第3章 使用数据

“是相反的，” Tweedledee 继续说道，“逻辑就是，如果之前是这样，那之后就有可能；如果之前可能，那之后就一定是这样；但是因为不是这样，所以也不会这样。”^①

——刘易斯·卡罗尔

^① 出自《爱丽丝镜中奇遇记》，作者刘易斯·卡罗尔。公共领域。

3.1 创建和重定义变量

如果要编制一个 SAS 软件最受欢迎的功能列表，创建和重定义变量一定会名列前茅。幸运的是，SAS 的灵活性使其可用常规方法完成这些任务。你能用以下基本形式的赋值语句来创建和重定义变量：

```
variable = expression;
```

等号左侧是变量名，可以是新变量或已有变量。等号右侧可能会出现常量、另一个变量或数学表达式。以下是基本类型赋值语句的示例：

表达式类型	赋值语句
数值常量	<code>Qwerty = 10;</code>
字符常量	<code>Qwerty = 'ten';</code>
变量	<code>Qwerty = OldVar;</code>
加法	<code>Qwerty = OldVar + 10;</code>
减法	<code>Qwerty = OldVar - 10;</code>
乘法	<code>Qwerty = OldVar * 10;</code>
除法	<code>Qwerty = OldVar / 10;</code>
求幂	<code>Qwerty = OldVar ** 10;</code>

变量 `Qwerty` 是数值还是字符取决于定义它的表达式。当表达式是数值型时，`Qwerty` 是数值；当表达式是字符型时，`Qwerty` 是字符。

在决定如何解释表达式时，SAS 遵循标准的数学运算优先级法则：SAS 首先进行求幂，然后进行乘法和除法，最后进行加法和减法。你可以使用括号来改变这种运算顺序。通过使用括号，可以使以下两个看似相近的 SAS 语句产生截然不同的运算结果：

赋值语句	结果
<code>x = 10 * 4 + 3 ** 2;</code>	<code>x = 49</code>
<code>x = 10 * (4 + 3 ** 2);</code>	<code>x = 130</code>

虽然 SAS 可以读取带括号或不带括号的表达式，但是人却很难读懂复杂表达式。如果使用括号，你的程序将会更容易阅读。

示例 以下原始数据来自于对菜农的调查，菜农们被要求估计以下四种农作物收获

的产量（单位：磅）：西红柿、西葫芦、豌豆和葡萄。

```
Gregor 10 2 40 0
Molly 15 5 10 1000
Luther 50 10 15 50
Susan 20 0 . 20
```

以下程序从文件 Garden.dat 中读取数据，然后进行修改：

```
*使用赋值语句修改数据集homegarden;
DATA homegarden;
  INFILE 'c:\MyRawData\Garden.dat';
  INPUT Name $ 1-7 Tomato Zucchini Peas Grapes;
  Zone = 14;
  Type = 'home';
  Zucchini = Zucchini * 10;
  Total = Tomato + Zucchini +Peas + Grapes;
  PerTom = (Tomato / Total) * 100;
RUN;
PROC PRINT DATA = homegarden;
  TITLE 'Home Gardening Survey';
RUN;
```

该程序包含五个赋值语句。首先，创建新变量 Zone 并赋值数值常量 14。新变量 Type 被赋值字符常量 home。变量 Zucchini 等于 Zucchini 乘以 10，因为这样看起来才像西葫芦的重量。变量 Total 等于四类农作物重量之和。变量 PerTom 不是西红柿的重量，而是西红柿的重量占比。PROC PRINT 输出的报表包含所有已有的和新生成的变量，见表 3-1。

表 3-1 菜农调查报表

Obs	Name	Tamato	Zucchini	Peas	Grapes	Zone	Type	Total	PerTom
1	Gregor	10	20	40	0	14	home	70	14.2857
2	Molly	15	50	10	1000	14	home	1075	1.3953
3	Luther	50	100	15	50	14	home	215	23.2558
4	Susan	20	0	.	20	14	home	.	.

请注意，变量 Zucchini 只出现一次，因为新值替换了旧值。其他四个赋值语句均创建了新变量。当变量是新变量时，SAS 将它添加到正在创建的数据集中。若变量已经存在，SAS 则使用新值替换原始值。使用已有变量的优点是，不会使你的数据集因有许多相似变量而混乱。但是，一般不建议你覆盖已有变量，除非确定不再需要该变量的原始值。

变量 Peas 的最后一个观测有缺失值。因此，该观测由 Peas 计算得到的变量 Total 和 PerTom 也被设置为缺失值，并且日志中出现以下消息：

NOTE：缺失值的生成是对缺失值执行操作的结果。

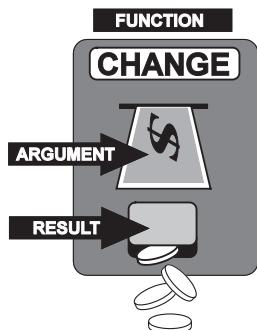
该消息通常提示你这儿有一个错误。然而，在上述情况中，它并不是错误，而只是数据收集不完整的结果^①。

3.2 使用 SAS 函数

有时候，只使用算术运算符的简单表达式并不能让你得到想要的新值。这时就要用到灵巧的函数了，函数可以简化你的任务，因为 SAS 已经替你完成了程序设计。你所需要做的只是将正确的值输入函数当中，你就能得到想要的结果，这就好比如图把一美元

投入零钱兑换机取回四个 25 美分硬币一样便捷。

SAS 有数百个函数，常用领域包括：



字符	宏
字符串匹配	数学
日期和时间	概率
描述性统计量	随机数
距离	州和邮编
金融	可变信息

接下来的两节将给出最常用的 SAS 函数的示例。函数对函数名后面括号中的参数进行计算或转换。SAS 函数有以下基本格式：

```
function-name(argument, argument, ...)
```

即便没有任何参数，所有函数都必须带括号。参数之间用逗号分隔，参数可以是变量名称、数字、带引号的字符常量或者是表达式。以下语句使用函数 MDY() 与变量 MonthBorn, DayBorn 和 YearBorn 计算 SAS 日期型变量 Birthday 的值。MDY() 函数有三个参数，分别为月、日和年：

```
Birthday = MDY(MonthBorn, DayBorn, YearBorn);
```

① 如果你只想累加非缺失值，可以使用第 11.7 节中讨论的 SUM 函数来实现。

函数可以嵌套使用，即一个函数（的处理结果）是另一个函数的参数。例如，以下语句使用两个嵌套函数 INT() 和 LOG() 来计算 NewValue 的值：

```
NewValue = INT(LOG(10));
```

此示例的结果为 2，取数字常量 10 的自然对数（2.3026）的整数部分。在使用嵌套函数时要注意每个括号都是成对出现的。

示例 下面示例用来自南瓜雕刻竞赛的数据来说明几种函数的使用。变量的顺序依次为：参赛者姓名，参赛者年龄，南瓜类型（雕刻或装饰），报名日期和五名评委给出的分数：

	Alicia Grossman	13	c	10-28-2012	7.8	6.5	7.2	8.0	7.9
Matthew Lee		9	D	10-30-2012	6.5	5.9	6.8	6.0	8.1
Elizabeth Garcia		10	C	10-29-2012	8.9	7.9	8.5	9.0	8.8
Lori Newcombe		6	D	10-30-2012	6.7	5.6	4.9	5.2	6.1
Jose Martinez		7	d	10-31-2012	8.9	9.5	10.0	9.7	9.0
Brian Williams		11	C	10-29-2012	7.8	8.4	8.5	7.9	8.0

以下程序读取这些数据，创建两个新变量（AvgScore 和 DayEntered）并转换另一个变量（Type）：

```
DATA contest;
  INFILE 'c:\MyRawData\Pumpkin.dat';
  INPUT Name $16. Age 3. +1 Type $1. +1 Date MMDDYY10.
    (Scr1 Scr2 Scr3 Scr4 Scr5) (4.1);
  AvgScore = MEAN(Scr1, Scr2, Scr3, Scr4, Scr5);
  DayEntered = DAY(Date);
  Type = UPCASE(Type);
RUN;
PROC PRINT DATA = contest;
  TITLE 'Pumpkin Carving Contest';
RUN;
```

变量 AvgScore 使用 MEAN() 函数创建，该函数返回所有非缺失值参数的平均值。这与简单地将参数加在一起并除以它们的个数不同，如果有任何参数为缺失值，后者将返回一个缺失值。

变量 DayEntered 是使用 DAY() 函数创建的，该函数返回日期在月中的第几天。SAS 具有各种操作日期的函数，而使用它们最大的好处就是你不用担心像闰年这样的事情，SAS 已经事先为你考虑了。

变量 Type 使用 UPCASE() 函数进行转换。当涉及变量值时，SAS 区分大小写；“d”与“D”不一样。数据文件中变量 Type 的值既有小写字母也有大写字母，因此 UPCASE() 函数用于使所有字母都转换为大写。

结果如表 3-2 所示。

表 3-2 南瓜雕刻竞赛数据

Obs	Name	Age	Type	Date ^①	Scr1	Scr2	Scr3	Scr4	Scr5	AvgScore	DayEntered
1	Alicia Grossman	13	C	19294	7.8	6.5	7.2	8.0	7.9	7.48	28
2	Matthew Lee	9	D	19296	6.5	5.9	6.8	6.0	8.1	6.66	30
3	Elizabeth Garcia	10	C	19295	8.9	7.9	8.5	9.0	8.8	8.62	29
4	Lori Newcombe	6	D	19296	6.7	5.6	4.9	5.2	6.1	5.70	30
5	Jose Martinez	7	D	19297	8.9	9.5	10.0	9.7	9.0	9.42	31
6	Brian Williams	11	C	19295	7.8	8.4	8.5	7.9	8.0	8.12	29

3.3 常用 SAS 字符函数

表 3-3 常用 SAS 字符函数表

函 数 名	语 法 ^②	定 义
字符		
ANYALNUM	ANYALNUM(arg,start)	返回首次出现任意字母或数字的位置，可选择起始查找位置。
ANYALPHA	ANYALPHA(arg,start)	返回首次出现任意字母的位置，可选择起始查找位置。
ANYDIGIT	ANYDIGIT(arg,start)	返回首次出现任意数字的位置，可选择起始查找位置。
ANYSPACE	ANYSPACE(arg,start)	返回首次出现空格的位置，可选择起始查找位置。
CAT	CAT(arg-1,arg-2,...arg-n)	连接两个或多个字符串，保留首尾全部空格。
CATS	CATS(arg-1,arg-2,...arg-n)	连接两个或多个字符串，移除首尾全部空格。
CATX	CATX('separator-string', arg-1,arg-2,...arg-n)	连接两个或多个字符串，移除首尾全部空格，并在各字符串之间插入指定的分隔符。
COMPRESS	COMPRESS(arg, 'char')	移除字符串中的空格或可选字符。

① 请注意，这些打印出来的日期是自 1960 年 1 月 1 日以来的天数。第 4.6 节将讨论如何把这些日期值格式化成可读取的日期。

② arg 是参数的缩写，可代表文字值、变量名或表达式。

(续表)

函数名	语 法	定 义
INDEX	INDEX(arg, 'string')	返回字符串的起始位置。
LEFT	LEFT(arg)	将字符串左对齐。
LENGTH	LENGTH(arg)	返回字符串长度, 不考虑尾部空格(缺失值长度为 1)。
PROPCASE	PROPCASE(arg)	将单词中首字母转换成大写字母, 其余字母转换成小写字母。
SUBSTR	SUBSTR(arg,position,n)	从 position 位置开始提取长度为 n 的子串; 若未指定长度 n, 则提取至字符串末尾 ^① 。
TRANSLATE	TRANSLATE(source,to-1,from-1,...to-n,from-n)	将 from 字符串和 to 字符串中的字符一一对应后, 用 to 字符替换 source 中的 from 字符(仅能一对一字符替换, 例如不能用两个字符替换一个字符)。
TRANWRD	TRANWRD(source,from,to)	将 source 字符串中包含的 from 字符串替换为 to 字符串。
TRIM	TRIM(arg)	移除字符串尾部空格。
UPCASE	UPCASE(arg)	把参数中所有字母转换成大写。

表 3-4

函 数 名	示 例	结 果	示 例	结 果
字符				
ANYALNUM	a='123 E St, #2'; x=ANYALNUM(a);	x=1	a='123 E St, #2'; y=ANYALNUM(a,10);	y=12
ANYALPHA	a='123 E St, #2'; x=ANYALPHA(a);	x=5	a='123 E St, #2'; y=ANYALPHA(a,10);	y=0
ANYDIGIT	a='123 E St, #2'; x=ANYDIGIT(a);	x=1	a='123 E St, #2'; y=ANYDIGIT(a,10);	y=12
ANYSPACE	a='123 E St, #2'; x=ANYSPACE(a);	x=4	a='123 E St, #2'; y=ANYSPACE(a,10);	y=10
CAT	a=' cat';b='dog'; x=CAT(a,b);	x='catdog'	a='cat';b=' dog'; y=CAT(a,b);	y='cat dog'
CATS	a=' cat';b='dog'; x=CATS(a,b);	x='catdog'	a='cat';b=' dog'; y=CATS(a,b);	y='catdog'
CATX	a=' cat';b='dog'; x=CATX('&',a,b);	x='cat dog'	a=' cat';b='dog'; y=CATX('&',a,b);	y='cat&dog'
COMPRESS	a=' cat & dog'; x=COMPRESS(a);	x='cat&dog'	a=' cat & dog'; y=COMPRESS(a,'&');	y='cat dog'

① 当 SUBSTR 在等号的左侧时, 其具有不同的功能。

(续表)

函数名	示例	结果	示例	结果
INDEX	a='123 E St, #2'; x=INDEX(a,'#');	x=11	a='123 E St, #2'; y=INDEX(a,'St');	y=7
LEFT	a=' cat'; x=LEFT(a);	x='cat'	a=' m y c a t ' ; y=LEFT(a);	y='my cat'
LENGTH	a = ' m y c a t ' ; x=LENGTH(a);	x=6	a = ' m y c a t ' ; y=LENGTH(a);	y=7
PROPCASE	a = ' M y C a t ' ; x=PROPCASE(a);	x='Mycat'	a = ' T I G E R ' ; y=PROPCASE(a);	y='Tiger'
SUBSTR	a='(916)734-6281'; x=SUBSTR(a,2,3);	x='916'	y=SUBSTR('1cat',2);	y='cat'
TRANSLATE	a = ' 6 / 1 6 / 9 9 ' ; x=TRANSLATE (a,'-','/');	x='6-16-99'	a='my cat can'; y=TRANSLATE (a, 'r','c');	y='my rat ran'
TRANWRD	a='Main Street'; x=TRANWRD (a,'Street','St');	x='Main St'	a='my cat can'; y=TRANWRD (a, 'cat','rat');	y='my rat can'
TRIM	a='my'; b='cat'; x=TRIM(a) b; ^①	x='mycat'	a='my cat'; b='s'; y=TRIM(a) b;	y='my cats'
UPCASE	a = ' M y C a t ' ; x=UPCASE(a);	x='MYCAT'	y=UPCASE('Tiger');	y='TIGER'

3.4 常用 SAS 数值函数

表 3-5 常用 SAS 数值函数表

函数名	语 法 ^②	定 义
数值		
INT	INT(arg)	返回参数的整数部分
LOG	LOG(arg)	返回自然对数值
LOG10	LOG10(arg)	返回底数为 10 的对数值
MAX	MAX(arg-1,arg-2,...arg-n)	返回最大的非缺失值
MEAN	MEAN(arg-1,arg-2,...arg-n)	返回非缺失值的算术平均值

① 连接运算符 || 可以连接字符串。

② arg 是参数的缩写，可代表文字值，变量名或表达式。

(续表)

函 数 名	语 法	定 义
MIN	MIN(arg-1,arg-2,...arg-n)	返回最小的非缺失值
N	N(arg-1,arg-2,...arg-n)	返回非缺失值的个数
NMISS	NMISS(arg-1,arg-2,...arg-n)	返回缺失值的个数
ROUND	ROUND(arg, round-off-unit)	返回按指定精度四舍五入后的值
SUM	SUM(arg-1,arg-2,...arg-n)	返回非缺失值的和
日期		
DATEJUL	DATEJUL(julian-date)	将儒略日期 (Julian date) 转换成 SAS 日期值 ^①
DAY	DAY(date)	返回 SAS 日期值中所在月份的第几天
MDY	MDY(month,day,year)	返回由月、日、年生成的 SAS 日期值
MONTH	MONTH(date)	返回 SAS 日期值中的月份 (1-12)
QTR	QTR(date)	返回 SAS 日期值中的季度 (1-4)
TODAY	TODAY()	返回当天 SAS 日期值
WEEKDAY	WEEKDAY(date)	返回 SAS 日期值所在周的第几天 (1 = 星期天)
YEAR	YEAR(date)	返回 SAS 日期值中的年份
YRDIF	YRDIF(start-date,end- date,'AGE')	计算两个 SAS 日期值间隔的年数，闰年考虑在内

表 3-6

函 数 名	示 例	结 果	示 例	结 果
数值				
INT	x=INT(4.32);	x=4	y=INT(5.789);	y=5
LOG	x=LOG(1);	x=0.0	y=LOG(10);	y=2.30259
LOG10	x=LOG10(1);	x=0.0	y=LOG10(10);	y=1.0
MAX	x=MAX(9.3,8,7.5);	x=9.3	y=MAX(-3,,,5);	y=5
MEAN	x=MEAN(1,4,7,2);	x=3.5	y=MEAN(2,,,3);	y=2.5
MIN	x=MIN(9.3,8,7.5);	x=7.5	y=MIN(-3,,,5);	y=-3
N	x=N(1,,,7,2);	x=3	y=N(,,4,,,);	y=1
NMISS	x=NMISS(1,,,7,2);	x=1	y=NMISS(,,4,,,);	y=3

^① SAS 日期值是自 1960 年 1 月 1 日起的天数。

(续表)

函 数 名	示 例	结 果	示 例	结 果
ROUND	x=ROUND(12.65);	x=13	y=ROUND(12.65,.1);	y=12.7
SUM	x=SUM(3,5,1);	x=9.0	y=SUM(4,7,);	y=11
日期				
DATEJUL	a=60001; x=DATEJUL(a);	x=0	a=60365; y=DATEJUL(a);	y=364
DAY	a=MDY(4,18,2012); x=DAY(a);	x=18	a=MDY(9,3,60); y=DAY(a);	y=3
MDY	x=MDY(1,1,1960);	x=0	m=2; d=1; y=60; Date=MDY(m,d,y);	Date=31
MONTH	a=MDY(4,18,2012); x=MONTH(a);	x=4	a=MDY(9,3,60); y=MONTH(a);	y=9
QTR	a=MDY(4,18,2012); x=QTR(a);	x=2	a=MDY(9,3,60); y=QTR(a);	y=3
TODAY	x=TODAY();	x=today's date	y=TODAY()-1;	y=yesterday's date
WEEKDAY	a=MDY(4,13,2012); x=WEEKDAY(a);	x=6	a=MDY(4,18,2012); y=WEEKDAY(a);	y=4
YEAR	a=MDY(4,13,2012); x=YEAR(a);	x=2012	a=MDY(1,1,1960); y=YEAR(a);	y=1960
YRDIF	a=MDY(4,13,2000); b=MDY(4,13,2012); x=YRDIF(a,b,'AGE');	x=12.0	a=MDY(4,13,2000); b=MDY(8,13,2012); y=YRDIF(a,b,'AGE');	y=12.3342

3.5 使用 IF-THEN 语句

通常，你会希望只对部分观测而不是全部观测进行赋值，在满足某些条件时赋值而其他条件时不赋值。这称为条件逻辑，你可以用 IF-THEN 语句来实现，其基本形式如下：

```
IF condition THEN action;
```

其中，条件 (condition) 是将一个事物与另一个事物进行比较的表达式，当该表达式为真时，SAS 会执行后面的动作 (action) 语句，通常为一个赋值语句。例如

```
IF Model = 'Berlinetta' THEN Make = 'Ferrari';
```

该语句告诉 SAS 当变量 Model 等于 “Berlinetta” 时，设置变量 Make 等于 “Ferrari”。条件两边的比较项可以是常量、变量或表达式。这些比较项通过比较运算符分隔，比较运算符可以是符号或助记符。使用符号还是助记符取决于你的个人偏好和键盘上可用的符号。以下是基本的比较运算符：

符号	助记符	含义
=	EQ	等于
^ =, ^ =, 或 ~ =	NE	不等于
>	GT	大于
<	LT	小于
> =	GE	大于等于
< =	LE	小于等于

IN 运算符也可以进行比较，但是它的用法有点不同。IN 将变量值与值列表进行比较。示例如下：

```
IF Model IN ('Model T', 'Model A') THEN Make = 'Ford';
```

该语句告诉 SAS，只要变量 Model 的值是 “Model T” 或 “Model A”，设置变量 Make 等于 “Ford”。

一个 IF-THEN 语句只能有一个动作。如果你添加关键字 DO 和 END，则可以执行多个动作。例如：

```
IF condition THEN DO;
  action;
  action;
END;                                IF Model = 'DMC-12' THEN DO;
                                         Make = 'DeLorean';
                                         BodyStyle = 'coupe';
                                         END;
```

DO 语句将在其后出现直至与之匹配的 END 语句为止的所有 SAS 语句视为一个单元。DO 语句，END 语句和它们之间的所有语句一起被称为 DO 组合。

你还可以使用关键字 AND 和 OR 来指定多个条件：

```
IF condition AND condition THEN action;
```

例如：

```
IF Make = 'Alfa Romeo' AND Model = 'Tipo B' THEN Seats = 1;
```

与比较运算符一样，AND 和 OR 也可以是符号或助记符：

符号	助记符	含义
&	AND	所有的表达式必须为真
, , 或 !	OR	至少有一个表达式为真

使用长的比较字符串时要注意，因为它们可能成为一个逻辑迷宫。

示例 以下数据显示了在拍卖中出售的珍奇古董车信息。数据值依次为制造厂商、型号、汽车制造年份、座位数量和销售价格（以百万美元为单位）：

DeDion	LaMarquise	1884	4	4.6
Rolls-Royce	Silver Ghost	1912	4	1.7
Mercedes-Benz	SSK	1929	2	7.4
	F-88	1954	.	3.2
Ferrari	250 Testa Rossa	1957	2	16.3

以下程序从文件 Auction.dat 中读取数据，并使用 IF-THEN 语句。

```

DATA oldcars;
  INFILE 'c:\MyRawData\Auction.dat';
  INPUT Make $ 1-13 Model $ 15-29 YearMade Seats MillionsPaid;
  IF YearMade < 1890 THEN Veteran = 'Yes';
  IF Model = 'F-88' THEN DO;
    Make = 'Oldsmobile';
    Seats = 2;
  END;
RUN;
PROC PRINT DATA = oldcars;
  TITLE 'Cars Sold at Auction';
RUN;

```

该程序包含两个 IF-THEN 语句。1890 年前制造的汽车被分类为 Veteran。第一个 IF-THEN 创建一个名为 Veteran 的新变量，并对所有 1890 年前制造的汽车赋值 Yes。第二个 IF-THEN 使用 DO 和 END 给型号为 F-88 的汽车填充缺失的数据。输出如表 3-7 所示。

表 3-7 汽车拍卖销售报表

Obs	Make	Model	YearMade	Seats	MillionsPaid	Veteran
1	DeDion	LaMarquise	1884	4	4.6	Yes
2	Rolls-Royce	Silver Ghost	1912	4	1.7	

(续表)

Obs	Make	Model	YearMade	Seats	MillionsPaid	Veteran
3	Mercedes-Benz	SSK	1929	2	7.4	
4	Oldsmobile	F-88	1954	2	3.2	
5	Ferrari	250 Testa Rossa	1957	2	16.3	

3.6 用 IF-THEN/ELSE 语句分组观测

使用 IF-THEN 语句的常见用途是将观测进行分组。例如，你可能有每日的数据，但需要季度报表，或者你可能有每次人口普查的数据，但希望按州进行分析。对数据进行分组有很多可能的原因，所以迟早你会有此类需求。

创建分组变量的最简单和最常见的方法是使用一系列 IF-THEN 语句。^① 通过在 IF 语句中添加关键字 ELSE，你可以告诉 SAS 这些语句是相关的。

IF-THEN/ELSE 逻辑采用以下基本形式：

```
IF condition THEN action;
ELSE IF condition THEN action;
ELSE IF condition THEN action;
```

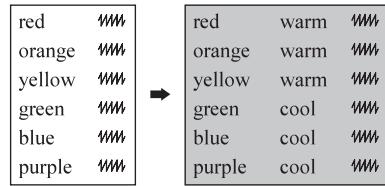
请注意，ELSE 语句只是在 IF-THEN 语句前面加上 ELSE。这样的语句你可以添加任意多个。

与不带任何 ELSE 语句的简单 IF-THEN 语句相比，IF-THEN/ELSE 逻辑有两个优点。首先，它的运算时间更少，效率更高，一旦观测满足某条件，SAS 就会跳过而不执行该系列其余语句。其次，ELSE 逻辑确保你的分组是相互排斥的，因此你不会错误地将一个观测分配到多个分组中。

有时系列中的最后一个 ELSE 语句有一点不同，它只包含一个动作而没有 IF 或 THEN。请注意系列中最后的 ELSE 语句：

```
IF condition THEN action;
ELSE IF condition THEN action;
```

^① 创建分组变量的其他方法包括使用 SELECT 语句，或使用 PROC FORMAT 生成的用户自定义格式的 PUT 函数。



```
ELSE action;
```

这种 ELSE 充当默认值的角色，它将自动执行所有未能满足任何先前的 IF 语句的观测。你只能有一个这样的语句，它必须是 IF-THEN/ELSE 系列中的最后一个语句。

示例 以下数据来自于家庭改善的调查。每个记录包含三个数据值：所有者姓名，所完成工作的描述和改善成本（美元）：

Bob	kitchen cabinet face-lift	1253.00
Shirley	bathroom addition	11350.70
Silvia	paint exterior	.
Al	backyard gazebo	3098.63
Norm	paint interior	647.77
Kathy	second floor addition	75362.93

以下程序从文件 Home.dat 中读取原始数据，然后新建一个名为 CostGroup 的分组变量。此变量被赋予 high、medium、low 或 missing 值，具体取决于 Cost 的值：

```
*对观测按照变量cost分组;
DATA homeimprovements;
  INFILE 'c:\MyRawData\Home.dat';
  INPUT Owner $ 1-7 Description $ 9-33 Cost;
  IF Cost = . THEN CostGroup = 'missing';
  ELSE IF Cost < 2000 THEN CostGroup = 'low';
  ELSE IF Cost < 10000 THEN CostGroup = 'medium';
  ELSE CostGroup = 'high';
RUN;
PROC PRINT DATA = homeimprovements;
  TITLE 'Home Improvement Cost Groups';
RUN;
```

请注意，此 IF-THEN/ELSE 系列中有四个语句，分别赋予变量 CostGroup 不同的值。第一个语句处理变量 Cost 为缺失数据的观测。如果没有这个语句，Cost 为缺失值的观测将被错误地分配给 CostGroup 为 low 的组。SAS 认为缺失值小于非缺失值，小于任何可打印字符，也小于数值变量中的负数。除非你确定数据不包含缺失值，否则在写 IF-THEN/ELSE 语句时应考虑缺失值情况。

结果如表 3-8 所示。

表 3-8 家庭改善成本分组

Obs	Owner	Description	Cost	CostGroup
1	Bob	kitchen cabinet face-lift	1253.00	low
2	Shirley	bathroom addition	11350.70	high
3	Silvia	paint exterior	.	missing
4	Al	backyard gazebo	3098.63	medium
5	Norm	paint interior	647.77	low
6	Kathy	second floor addition	75362.93	high

3.7 提取数据的子集

通常，程序员发现他们想要使用数据集里的某些观测而排除其余观测。最常见的方法是在 DATA 步中使用取子集的 IF 语句^①。其基本形式是：

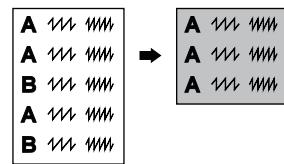
```
IF expression;
```

例如：

```
IF Sex = 'f';
```

乍一看提取子集的 IF 语句可能看起来很奇怪。人们自然会问：“IF Sex = ‘f’，接下来要干什么？”提取子集 IF 语句看起来不完整，就像是粗心大意的打字员按下删除键太久一样。但这确实是标准 IF-THEN 语句的一个特例。在这种情况下，动作是隐含的。如果表达式为真，则 SAS 继续执行 DATA 步；如果表达式为假，则停止处理该观测的后续语句，该观测不会添加到正在创建的数据集，然后 SAS 接着处理下一个观测。你可以将取子集 IF 语句视为一种开关切换。如果条件为真，则开关打开，观测被处理；如果条件为假，则开关关闭，观测不会被处理。

如果你不喜欢取子集 IF 语句，还可以选择用 DELETE 语句。DELETE 语句与取子集 IF 语句正好相反。取子集 IF 语句告诉 SAS 要包含哪些观测，而 DELETE 语句告诉 SAS 哪些观测要排除：



^① 有关取子集 IF 语句的更多信息，请参见第 2.13 节和第 6.9 节。对数据取子集的其他方法还包括 WHERE 语句（第 4.2 节和附录）和“WHERE=” 选项（第 6.13 节）。

```
IF expression THEN DELETE;
```

以下两个语句是等效的（假设变量 Sex 只包含两个值，并且没有缺失数据）：

```
IF Sex = 'f';  IF Sex = 'm' THEN DELETE;
```

示例 当地业余剧场的演员想在今年春天的戏剧演出中表演莎士比亚的喜剧。你志愿使用在线百科全书来编制一个标题列表。对于每场演出，你的数据文件包含：戏剧标题，首演年份和戏剧类型：

A Midsummer Night's Dream	1595	comedy
Comedy of Errors	1590	comedy
Hamlet	1600	tragedy
Macbeth	1606	tragedy
Richard III	1594	history
Romeo and Juliet	1596	tragedy
Taming of the Shrew	1593	comedy
Tempest	1611	romance

以下程序从 Shakespeare.dat 的原始数据文件中读取数据，然后使用取子集 IF 语句从中仅选择喜剧：

```
*只选取comedies类型;
DATA comedy;
  INFILE 'c:\MyRawData\Shakespeare.dat';
  INPUT Title $ 1-26 Year Type $;
  IF Type = 'comedy';
RUN;
PROC PRINT DATA = comedy;
  TITLE 'Shakespearean Comedies';
RUN;
```

输出结果如表 3-9 所示。

表 3-9 莎士比亚喜剧

Obs	Title	Year	Type
1	A Midsummer Night's Dream	1595	comedy
2	Comedy of Errors	1590	comedy
3	Taming of the Shrew	1593	comedy

日志中出现了下面的注释，表示尽管从输入文件中读取了 8 条记录，但数据集 WORK.COMEDY 只包含 3 个观测：

NOTE: 从 INFILE 'c:\MyRawData\Shakespeare.dat' 中读取了 8 条记录。
NOTE: 数据集 WORK.COMEDY 有 3 个观测和 3 个变量。

当你对观测取子集时，检查 SAS 日志总是个好主意，它能确保生成的结果是想要的。在上面的程序中，你可以用语句

```
IF Type = 'tragedy' OR Type = 'romance' OR Type = 'history' THEN  
DELETE;
```

替换以下语句

```
IF Type = 'comedy';
```

但是你不得不打更多的字。通常当更容易指定条件包含观测时，使用取子集 IF 语句，而在更容易指定条件排除观测时，使用 DELETE 语句。

3.8 使用 SAS 日期

日期使用起来可能很棘手。有的月份是 30 天，有的月份是 31 天，有的月份是 28 天，而且不要忘记闰年。SAS 日期简化了这一切。SAS 日期是等于自 1960 年 1 月 1 日以来的天数值。下表列出了 4 个日期及其作为 SAS 日期的取值：

日期	SAS 日期值	日期	SAS 日期值
January 1, 1959	-365	January 1, 1961	366
January 1, 1960	0	January 1, 2020	21915

SAS 具有处理日期的专用工具：读取日期的输入格式，操作日期的函数以及打印日期的输出格式。^①下一章节将用表格显示常用的日期输入格式、输出格式和函数。

输入格式 要读取日期型变量，你可以使用格式化的输入方式。SAS 有许多以不同形式读取日期的日期输入格式。所有这些输入格式将你的日期数据转换为自 1960 年 1 月 1 日以来的天数。下面的 INPUT 语句告诉 SAS 使用 ANYDTDTE9 输入格式^② 读取

^① SAS 还具有处理 time 值（自午夜以来的秒数）的输入格式，函数和输出格式，以及 datetime 值（自 1960 年 1 月 1 日午夜开始的秒数）。

^② ANYDTDTEw. 是一个特殊的输入格式，可以读取几乎任何形式的日期。如果日期不明确，例如 01-02-03，则 SAS 使用 DATESTYLE= 系统选项的值来确定月、日、年的顺序。DATESTYLE= 的默认值为 MDY (月, 日, 年)。

BirthDate 变量。

```
INPUT BirthDate ANYDTDTE9.;
```

设置输入的默认世纪 当 SAS 看到像 07/04/76 两位数年份的日期时，SAS 必须决定该年份在哪个世纪，是 1976 年，2076 年，还是 1776 年？系统选项 “YEARCUTOFF=” 指定 SAS 使用的百年跨度的第一年。在写本书时，此选项的默认值为 1920。你可以使用 OPTIONS 语句更改此值。为避免出现问题，你可能希望在输入包含两位数年份的数据时指定 “YEARCUTOFF=” 选项。以下语句告诉 SAS 将两位数的日期解释为发生在 1950 年至 2049 年之间：

```
OPTIONS YEARCUTOFF = 1950;
```

SAS 表达式中的日期 一旦使用 SAS 日期输入格式来读取变量，它就可以像其他数值变量一样在算术表达式中使用。例如，如果图书馆书籍在三周后到期，你可以通过在借出日期之后加 21 天来得到到期日。

```
DueDate = CheckDate + 21;
```

你可以在 SAS 表达式中使用日期作为常量。用 DATEw. 输出格式书写日期（例如 01JAN60），然后添加引号并紧跟字母 D。下面的赋值语句创建一个名为 EarthDay14 的变量，等于 April 22, 2014 的 SAS 日期值。

```
EarthDay14 = '22APR2014'D;
```

函数 SAS 日期函数能执行一些灵巧操作。以下语句使用三个函数从变量 BirthDate 来计算 age。

```
CurrentAge = INT (YRDIF(BirthDate, TODAY(), 'AGE'));
```

使用 “AGE” 参数的 YRDIF 函数，计算变量 BirthDate 和当前日期（来自 TODAY 函数）之间的年数。然后 INT 函数取返回值的整数部分。

输出格式 如果你打印 SAS 日期值，SAS 将默认打印实际值——自 1960 年 1 月 1 日起的天数。由于该数值对大多数人来说没有什么意义，SAS 提供了多种输出格式用来以不同的格式打印日期。^① 以下 FORMAT 语句告诉 SAS 用 WORDDATE18. 输出格式来打印变量 BirthDate。

^① 有关输出格式的更多信息，请参见第 4.6 节。

```
FORMAT BirthDate WORDDATE18.;
```

示例 本地图书馆有一个数据文件，其中包含有关图书卡的详细信息。每条记录都包含持卡人姓名，出生日期，发卡日期，以及上一次借书的到期日。

A. Jones	1-1-60	9-15-96	18JUN12
R. Grandage	03/18/1988	31 10 2007	5jul2012
K. Kaminaka	052903	2012024	12-MAR-12

以下程序读取原始数据，然后通过当前日期减去 DueDate 来计算变量 DaysOverDue。计算持卡人的当前年龄。然后 IF 语句使用日期常量来识别 January 1, 2012 之后发行的图书卡。

```
DATA librarycards;
  INFILE 'c:\MyRawData\Library.dat' TRUNCOVER;
  INPUT Name $11. + 1 BirthDate MMDDYY10. +1 IssueDate ANYDTDTE10.
        DueDate DATE11. ;
  DaysOverDue = TODAY() - DueDate;
  CurrentAge = INT(YRDIF(BirthDate, TODAY(), 'AGE'));
  IF IssueDate > '01JAN2012'D THEN NewCard = 'yes';
RUN;
PROC PRINT DATA = librarycards;
  FORMAT IssueDate MMDDYY8. DueDate WEEKDATE17. ;
  TITLE 'SAS Dates without and with Formats';
RUN;
```

表 3-10 是 PROC PRINT 的输出。请注意，变量 BirthDate 并没有按日期输出格式打印，而 IssueDate 和 DueDate 使用了输出格式。因为使用 TODAY 函数计算 DaysOverDue 和 CurrentAge，所以它们的值将根据程序运行日期而改变。DaysOverDue 的值对于未来到期的书籍是负值。

表 3-10 以无输出格式和有输出格式显示的 SAS 日期

Obs	Name	BirthDate	IssueDate	DueDate	DaysOverDue	CurrentAge	NewCard
1	A. Jones	0	09/15/96	Mon, Jun 18, 2012	0	52	
2	R. Grandage	10304	10/31/07	Thu, Jul 5, 2012	-17	24	
3	K. Kaminaka	15854	01/24/12	Mon, Mar 12, 2012	98	9	yes

3.9 常用日期输入格式、函数和输出格式

表 3-11 常用日期输入格式表

输入格式	定 义	宽度范围	默认宽度
ANYDTDTEw.	读取各种的日期格式	5-32	9
DATEw.	读取日期格式 : <i>ddmmmyy</i> 或 <i>ddmmmyyyy</i>	7-32	7
DDMMYYw.	读取日期格式 : <i>ddmmyy</i> 或 <i>ddmmyyyy</i>	6-32	6
JULIANw.	读取儒略日期格式 : <i>yyddd</i> 或 <i>yyyyddd</i>	5-32	5
MMDDYYw.	读取日期格式 : <i>mmddyy</i> 或 <i>mmddyyyy</i>	6-32	6

表 3-12 常用日期函数表

函数	语 法	定 义
DATEJUL	DATEJUL(<i>julian-date</i>)	将儒略日期转换成 SAS 日期值 ^① 。
DAY	DAY(<i>date</i>)	返回 SAS 日期值中所在月份的天数。
MDY	MDY(<i>month,day,year</i>)	返回由月、日、年组成的 SAS 日期值。
MONTH	MONTH(<i>date</i>)	返回 SAS 日期值中的月份（1-12）。
QTR	QTR(<i>date</i>)	返回 SAS 日期值中的季度（1-4）。
TODAY	TODAY()	返回当天 SAS 日期值。
WEEKDAY	WEEKDAY(<i>date</i>)	返回 SAS 日期值所在周的星期几（1 = 星期天）。
YEAR	YEAR(<i>date</i>)	返回 SAS 日期值中的年份。
YRDIF	YRDIF(<i>start-date,end-date, ‘AGE’</i>)	计算两个 SAS 日期值间隔的年数，闰年考虑在内。

表 3-13 常用日期输出格式表

输出格式	定 义	宽度范围	默认宽度
DATEw.	以下面的格式输出 SAS 日期 : <i>ddmmmyy</i>	5-11	7
EURDFDDw.	以下面的格式输出 SAS 日期 : <i>dd.mm.yy</i>	2-10	8
JULIANw.	将 SAS 日期值输出成儒略日期	5-7	5
MMDDYYw.	以下面格式输出 SAS 日期 : <i>mmddyy</i> or <i>mmddyyyy</i>	2-10	8
WEEKDATEw.	以下面格式输出 SAS 日期 : <i>day-of-week, month-name dd, yy</i> 或 <i>yyyy</i>	3-37	29
WORDDATEw.	以下面格式输出 SAS 日期 : <i>month-name dd, yyyy</i>	3-32	18

① SAS 日期值是自 1960 年 1 月 1 日起的天数。

表 3-14

输入格式	输入数据	INPUT 语句	结果
ANYDTDTEw.	1jan1961 01/01/61	INPUT Day ANYDTDTE10.;	366 366
DATEw.	1jan1961	INPUT Day DATE10.;	366
DDMMYYw.	01.01.61 02/01/61	INPUT Day DDMMYY8.;	366 367
JULIANw.	61001	INPUT Day JULIAN7.;	366
MMDDYYw.	01-01-61	INPUT Day MMDDYY8.;	366

表 3-15

函数	示例	结果	示例	结果
DATEJUL	a=60001; x=DATEJUL(a);	x=0	a=60365; y=DATEJUL(a);	y=364
DAY	a=MDY(4,18,2012); x=DAY(a);	x=18	a=MDY(9,3,60); y=DAY(a);	y=3
MDY	x=MDY(1,1,1960);	x=0	m=2; d=1; y=60; Date=MDY(m,d,y);	Date=31
MONTH	a=MDY(4,18,2012); x=MONTH(a);	x=4	a=MDY(9,3,60); y=MONTH(a);	y=9
QTR	a=MDY(4,18,2012); x=QTR(a);	x=2	a=MDY(9,3,60); y=QTR(a);	y=3
TODAY	x=TODAY();	x=today's date	y=TODAY()-1;	y=yesterday's date
WEEKDAY	a=MDY(4,13,2012); x=WEEKDAY(a);	x=6	a=MDY(4,18,2012); y=WEEKDAY(a);	y=4
YEAR	a=MDY(4,13,2000); x=YEAR(a);	x=2000	a=MDY(1,1,1960); y=YEAR(a);	y=1960
YRDIF	a=MDY(4,13,2000); b=MDY(4,13,2012); x=YRDIF(a,b,'AGE');	x=12	a=MDY(4,13,2000); b=MDY(8,13,2012); y=YRDIF(a,b,'AGE');	y=12.3342

表 3-16

输出格式	输入数据	PUT 语句 ^①	结果
DATEw.	366	PUT Birth DATE7.; PUT Birth DATE9.;	01JAN61 01JAN1961
EURDFDDw.	366	PUT Birth EURDFDD8.; PUT Birth EURDFDD10.;	01.01.61 01.01.1961

① 输出格式可用于 DATA 步中的 PUT 语句和 PUT 函数，也可用于 DATA 或 PROC 步中的 FORMAT 语句中。

(续表)

输出格式	输入数据	PUT 语句	结果
JULIANw.	366	PUT Birth JULIAN5.; PUT Birth JULIAN7.;	61001 1961001
MMDDYYw.	366	PUT Birth MMDDYY6.; PUT Birth MMDDYY10.;	010161 01/01/1961
WEEKDATEw.	366	PUT Birth WEEKDATE9.; PUT Birth WEEKDATE29.;	Sunday Sunday, January 1, 1961
WORDDATEw.	366	PUT Birth WORDDATE12.; PUT Birth WORDDATE18.;	Jan 1, 1961 January 1, 1961

3.10 使用 RETAIN 语句与求和语句

当 SAS 读取原始数据时，在 DATA 步每次迭代开始时会将所有变量设置为缺失值。这些值可以通过 INPUT 语句或赋值语句来更改，但当 SAS 返回到 DATA 步的开始并处理下一个观测时，它们将被重新设置为缺失值。可以用 RETAIN 和求和语句改变这种方式。如果变量出现在 RETAIN 语句中，那么它的值将从 DATA 步的一次迭代保留到下一个迭代。求和语句不仅保留了上一次迭代中的值，还将其值加到表达式中。

RETAIN 语句 当你希望 SAS 从 DATA 步的上一次迭代中保留变量的值时，请使用 RETAIN 语句。RETAIN 语句可以出现在 DATA 步中的任何位置，它具有以下形式，其中在 RETAIN 关键字之后列出了所有要保留的变量：

```
RETAIN variable-list;
```

你还可以为变量指定初始值来代替缺失值。所有列在初始值之前的变量在 DATA 步第一次迭代时都将使用该初始值：

```
RETAIN variable-list initial-value;
```

求和语句 求和语句也保留了 DATA 步的上一次迭代的变量值，不同的是，它适用于将表达式的值累加到变量中这种特殊场景中。求和语句和赋值语句一样，不包含任何关键字。它基本形式如下：

```
variable + expression;
```

这里没有等号但也并不是输入错误。此语句将表达式的值累加到变量中，同时将变

量的值从 DATA 步的一个迭代保留到下一个迭代。该变量必须是数字，并且初始值为零。可以使用 RETAIN 语句和 SUM 函数重写此语句，如下所示：

```
RETAIN variable 0;  
variable = SUM(variable, expression);
```

正如你所看到的，求和语句的确是使用 RETAIN 的一个特例。

示例 以下展示了如何使用 RETAIN 和求和语句。小联盟棒球队 Walla Walla Sweets 的比赛数据分别是比赛日期、参赛球队、击球次数和跑垒次数：

6-19	Columbia Peaches	8	3
6-20	Columbia Peaches	10	5
6-23	Plains Peanuts	3	4
6-24	Plains Peanuts	7	2
6-25	Plains Peanuts	12	8
6-30	Gilroy Garlics	4	4
7-1	Gilroy Garlics	9	4
7-4	Sacramento Tomatoes	15	9
7-4	Sacramento Tomatoes	10	10
7-5	Sacramento Tomatoes	2	3

团队需要在其数据集里增加两个变量。一个显示本赛季的累计跑垒次数；另一个显示了迄今为止比赛中的最大跑垒次数。以下程序使用求和语句来计算累积跑垒次数，以及用 RETAIN 语句和 MAX 函数来确定迄今为止比赛中的最大跑垒次数：

```
*使用RETAIIN和求和语句来找出最大跑垒次数和总跑垒次数;  
DATA gamestats;  
  INFILE 'c:\MyRawData\Games.dat';  
  INPUT Month 1 Day 3-4 Team $ 6-25 Hits 27-28 Runs 30-31;  
  RETAIN MaxRuns;  
  MaxRuns = MAX(MaxRuns, Runs);  
  RunsToDate + Runs;  
RUN;  
PROC PRINT DATA = gamestats;  
  TITLE "Season's Record to Date";  
RUN;
```

变量 MaxRuns 被设置为等于其在 DATA 步上一次迭代值（因为它出现在 RETAIN 语句中）和变量 Runs 的值之中的最大值。变量 RunsToDate 将每次比赛的跑垒次数 Runs 加

到自身中，同时其值从 DATA 步的一次迭代保留到下一个迭代。这就产生了跑垒次数的累积记录。

结果如表 3-17 所示。

表 3-17 空分赛季记录

Obs	Month	Day	Team	Hits	Runs	MaxRuns	RunsToDate
1	6	19	Columbia Peaches	8	3	3	3
2	6	20	Columbia Peaches	10	5	5	8
3	6	23	Plains Peanuts	3	4	5	12
4	6	24	Plains Peanuts	7	2	5	14
5	6	25	Plains Peanuts	12	8	8	22
6	6	30	Gilroy Garlics	4	4	8	26
7	7	1	Gilroy Garlics	9	4	8	30
8	7	4	Sacramento Tomatoes	15	9	9	39
9	7	4	Sacramento Tomatoes	10	10	10	49
10	7	5	Sacramento Tomatoes	2	3	10	52

3.11 利用数组简化程序

有时候，你想对许多变量做同样的事情。例如，你可能希望对每个数值变量取对数，或者将每个出现的 0 更改为缺失值。你可以编写一系列赋值语句或 IF 语句，但如果你有很多变量需要转换，使用数组将简化并缩短你的程序。

数组是相似元素的有序集合，就像当地的购物中心有一组排列好的店铺可供选择一样。在 SAS 中，数组是一组变量。你可以将数组定义为喜欢的任何变量组，前提是它们要么全是数值，或者全是字符。这些变量可以是数据集中的现有变量，也可以是要创建的新变量。

数组使用 DATA 步中的 ARRAY 语句来定义。其基本形式如下：

```
ARRAY name (n) $ variable-list;
```

该语句中，name 为数组名称，n 是数组中的变量个数。在 (n) 的后面是变量名列表。列表中的变量个数必须等于括号中给出的数字（你也可以使用 {} 或 [] 来代替括号）。这

被称为显式数组，因为你显式声明了数组中的变量数。如果变量是字符且之前未定义，则需要加“\$”符号。

数组本身不与数据集一起存储，它仅在 DATA 步运行时存在。你可以给数组取任何名称，只要它不与数据集中的变量名或任何 SAS 关键字重名即可。数组的命名规则与变量的命名规则相同（不超过 32 个字符，以字母或下划线开头，后面可以为字母、数字或下划线）。

要使用数组名称引用变量，请给出该变量的数组名称和下标。变量列表中的第一个变量用下标 1，第二个用下标 2，依此类推。如果你有一个数组定义如下

```
ARRAY store (4) Macys Penneys Sears Target;
```

STORE (1) 指代变量 Macys；STORE (2) 指代变量 Penneys；STORE (3) 指代变量 Sears；STORE (4) 指代变量 Target。仅仅把数组定义好并不会为你带来任何好处，你要能够通过使用数组来把事情变得简单。

示例 KBRK 广播电台正在进行一项调查，调查要求听众给 5 首不同的歌曲进行评分。歌曲的评级分数为 1 ~ 5，其中 1 等于在播放歌曲时切换电台，5 等于播放歌曲时调高音量。如果听众没有听到这首歌，或者不在乎对这首歌的评论，那就可以输入 9。以下是收集的数据：

Albany	54	3	9	4	4	9
Richmond	33	2	9	3	3	3
Oakland	27	3	9	4	2	3
Richmond	41	3	5	4	5	5
Berkeley	18	4	4	9	3	2

列表是听众居住城市，听众年龄，以及听众对五首歌曲的评分。以下程序将所有 9 改为缺失值（变量名用歌曲名每个单词的首字母拼接命名）。

```
*将所有的9改为缺失值;  
DATA songs;  
  INFILE 'c:\MyRawData\KBRK.dat';  
  INPUT City $ 1-15 Age wj kt tr filp ttr;  
  ARRAY song (5) wj kt tr filp ttr;  
  DO i = 1 TO 5;  
    IF song(i) = 9 THEN song(i) = .;  
  END;
```

```

RUN;
PROC PRINT DATA = songs;
  TITLE 'KBRK Song Survey';
RUN;

```

数组 SONG 定义为由 5 个变量组成，它们出现在 INPUT 语句中，分别表示 5 首歌曲。接下来是一个迭代的 DO 语句，DO 语句和 END 语句之间的所有语句在这种情况下被执行 5 次，数组中的每个变量分别执行了 1 次。

变量 i 用作索引变量，每次通过 DO 循环递增 1。第一次通过 DO 循环，变量 i 的值为 1，IF 语句将读取 IF song(1) = 9 THEN song(1) = .;，这相当于 IF wj=9 THEN wj=.;。第二次通过 DO 循环，i 的值为 2，IF 语句将读取 IF song(2) = 9 THEN song(2) = .;，这相当于与 IF kt=9 THEN kt=.;。以此类推，处理完数组中的所有五个变量。

结果如表 3-18 所示。

表 3-18 KBRK 音乐调查

Obs	City	Age	wj	kt	tr	filp	ttr	i
1	Albany	54	3	.	4	4	.	6
2	Richmond	33	2	.	3	3	3	6
3	Oakland	27	3	.	4	2	3	6
4	Richmond	41	3	5	4	5	5	6
5	Berkeley	18	4	4	.	3	2	6

请注意，数组成员 SONG(1) 到 SONG(5) 没有输出到数据集中，但是变量 i 却输出了。你可以编写 5 个 IF 语句来代替使用数组并实现相同的结果。在这个程序中，这两种方法并没有很大区别，但是如果你的调查有 100 首歌曲而不是 5 首，那么使用数组显然是更好的方案。

3.12 使用变量名列表的快捷方式

在编写 SAS 程序时，你通常需要写一个变量名称列表。如果只有少数几个变量，可能不需要快捷方式。但是，如果要定义一个包含 100 个元素的数组，你输入了 49 个变量名，得知还要继续输入 51 个变量名，此时你或许会感到有些烦躁，甚至产生“一定存在

更简便的方法”的想法。事实上，的确有这样的方法。

你可以在几乎任何可以使用常规变量列表的地方使用变量名的缩写列表。在函数中，缩写列表之前必须有关键字 OF（例如，SUM(OF Cat8 - Cat12)）。否则，你只是将常规列表替换为缩写列表。

数字范围列表 以相同字符开始并以连续数字结尾的变量可以是数字范围列表的一部分。只要数字是顺序相连的，就能以任何数字开始和结束。例如，以下 INPUT 语句显示了一个变量列表及其缩写形式：

变量列表	缩写列表
INPUT Cat8 Cat9 Cat10 Cat11 Cat12;	INPUT Cat8 - Cat12;

名称范围列表 名称范围列表取决于 SAS 数据集中变量的内部顺序或位置。这取决于 DATA 步中变量的出现顺序。例如，给定以下 DATA 步，内部变量名顺序将是 Y A C H R B：

```
DATA example;
  INPUT y a c h r;
  b = c + r;
RUN;
```

要指定名称范围列表，先是第一个变量，接着是两个连字符，然后是最后一个变量。以下 PUT 语句使用名称范围来显示变量列表及其缩写形式：

变量列表	缩写列表
PUT y a c h r b;	PUT y -- b;

如果你不确定内部顺序，可以使用 PROC CONTENTS 的“POSITION”选项来查找。以下程序将列出永久 SAS 数据集 DISTANCE 中的变量，并按位置排序：

```
LIBNAME mydir 'c:\MySASLib';
PROC CONTENTS DATA = mydir.distance POSITION;
RUN;
```

在程序中包含名称范围列表时请小心。虽然它们可以节省输入时间，但也可能使你的程序变得更难以理解和调试。

名称前缀列表 以相同字符开头的变量可以是名称前缀列表的一部分，可以在某些 SAS 语句和函数中使用。例如：

变量列表

```
DogBills = SUM(DogVet,DogFood,Dog_Care);      DogBills = SUM(OF Dog:);
```

特殊 SAS 名称列表 特殊名称列表 _ALL_，_CHARACTER_ 和 _NUMERIC_ 也可以用于你想要的任何位置来表示 SAS 数据集中所有变量、所有字符变量或者所有数值变量。这些名称列表在你要计算某观测的所有数值变量的均值时 (MEAN (OF _NUMERIC_)) 或列出某观测的所有变量的值时 (PUT _ALL_;) 是有用的。

示例 无线电台 KBRK 想要修改上一节中的程序，该程序将所有的 9 更改为缺失值。现在，新程序不去更改原始变量，而是创建新的变量 (Song1 ~ Song5)，这些新变量将具有新的缺失值。新程序还使用 MEAN 函数计算平均分数。以下是数据：

Albany	54	3	9	4	4	9
Richmond	33	2	9	3	3	3
Oakland	27	3	9	4	2	3
Richmond	41	3	5	4	5	5
Berkeley	18	4	4	9	3	2

以下是新程序：

```
DATA songs;
  INFILE 'c:\MyRawData\KBRK.dat';
  INPUT City $ 1-15 Age wj kt      tr filp ttr;
  ARRAY new (5) Song1 - Song5;
  ARRAY old (5) wj -- ttr; DO i = 1 TO 5;
    IF old(i) = 9 THEN new(i) = .;
    ELSE new(i) = old(i);
  END;
  AvgScore = MEAN(OF Song1 - Song5);
PROC PRINT DATA = songs;
TITLE 'KBRK Song Survey';
RUN;
```

请注意，ARRAY 语句使用缩写变量列表，数组 NEW 使用数字范围列表，数组 OLD 使用名称范围列表。在迭代 DO 循环中，如果原始变量（数组 OLD）的值为 9，则将 Song 变量（数组 NEW）设置为缺失值。否则，它们被设置为等于原始值。在 DO 循环之后，使用函数 MEAN 中的缩写变量列表创建一个新变量 AvgScore。输出包括 OLD 数组 (wj ~ ttr) 和 NEW 数组 (Song1 ~ Song5) 的变量：

表 3-19 KBRK 音乐调查

Obs	City	Age	wj	kt	tr	filp	ttr	Song1	Song2	Song3	Song4	Song5	i	AvgScore
1	Albany	54	3	9	4	4	9	3	.	4	4	.	6	3.66667
2	Richmond	33	2	9	3	3	3	2	.	3	3	3	6	2.75000
3	Oakland	27	3	9	4	2	3	3	.	4	2	3	6	3.00000
4	Richmond	41	3	5	4	5	5	3	5	4	5	5	6	4.40000
5	Berkeley	18	4	4	9	3	2	4	4	.	3	2	6	3.25000

4

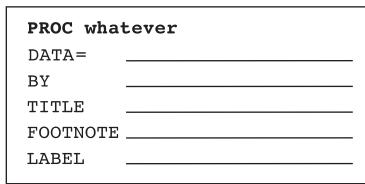
第4章 排序、打印和汇总数据

简单的事情一会儿便能马上解决。^①

——菲尔·加拉赫

^① 出自 SAS L Listserv，1994 年。经作者许可转载。

4.1 使用 SAS 过程



SAS 过程(即 PROC 步)的使用如同填写一张表格。别人设计出表格，而你只需要在空白处填写内容并从选项列表中进行选择即可。每个 PROC 步都有自己独特的组织形式以及选项列表。尽管如此，每个 SAS 过程也有相似的地方。本节主要讨论这些相似之处。

所有过程都包含必需的语句，而大多数过程也包含可选的语句。例如 PROC PRINT 就仅需要两个关键词：

```
PROC PRINT;
```

但是通过添加可选语句，你可以将该过程编写成十几行甚至更长。

PROC 语句 所有过程均以关键字 PROC 开头，后面跟着过程的名称，例如 PRINT 或 CONTENTS。若带有选项，则紧跟在过程名后。“DATA=” 选项告诉 SAS 使用哪个数据集作为该过程的输入。在以下示例中，SAS 将使用名为 BANANA 的临时 SAS 数据集：

```
PROC CONTENTS DATA = banana;
```

“DATA=” 选项显然是可选的。若你不添加该语句，SAS 将使用最新创建但不一定是最近使用过的数据集，作为该过程的输入。相比确定 SAS 默认所使用的数据集，直接指定你想使用的数据集会更加简便。若要使用永久 SAS 数据集，需要用 LIBNAME 语句，设置逻辑库引用名，指向你的数据集的存储位置，然后在 “DATA=” 选项中指定数据集的两级名称（如 2.19 节讨论）：

```
LIBNAME tropical 'c:\MySASLib';
PROC CONTENTS DATA = tropical.banana;
```

你也可以直接引用它，方法是将永久 SAS 数据集在操作环境中的名称放置在引号中（如 2.20 节讨论）。

```
PROC CONTENTS DATA = 'c:\MySASLib\banana';
```

BY 语句 BY 语句仅在 PROC SORT 过程中是必需的。在 PROC SORT 中，BY 语句告诉 SAS 如何排列观测。在所有其他过程中，BY 语句都是可选的，它告诉 SAS 为 BY 变量值的每种组合执行单独的分析，而不是将所有观测视为一个组。例如，以下语句

告诉 SAS 为每个州 (State) 运行单独的分析:

```
BY State;
```

除 PROC SORT 之外的所有过程均假定你的数据已按 BY 语句中的变量值排过序。若你的观测尚未排序，则使用 PROC SORT 进行排序。

TITLE 和 FOOTNOTE 语句 本书中已出现过很多次 TITLE 语句。FOOTNOTE 语句与之相似，只是它打印在页面底部。这些全局语句从技术上来说不是任何过程步的一部分。你可以将它们放置在程序中的任何位置，但由于它们应用于过程输出，通常将它们与某个过程放在一起会更有意义。最基本的 TITLE 语句由关键字 TITLE 以及其后包含在引号中的标题组成。SAS 不介意引号是单引号还是双引号，只要它们前后一致即可：

```
TITLE 'This is a title';
```

若你发现标题中含撇号，可以使用双引号将标题引起来或将单撇号替换为双撇号：

```
TITLE "Here's another title";
TITLE 'Here''s another title';
```

你最多可以指定 10 个标题或脚注，只需在关键字 TITLE 和 FOOTNOTE 后面加上数字即可：

```
FOOTNOTE3 'This is the third footnote';
```

标题和脚注会一直有效，除非你将它们替换为新的标题和脚注或是使用了空语句将它们取消。以下空语句会取消所有当前标题：

```
TITLE;
```

当你指定新标题或脚注时，它会替换具有相同数字的旧标题或脚注并取消更大数字的标题或脚注。例如，新的 TITLE2 会取消现有的 TITLE3（若存在）。

LABEL 语句 默认情况下，SAS 使用变量名作为输出的标签，但使用 LABEL 语句的话可以为每个变量创建更具说明性的标签，标签的长度最大可以达到 256 个字符。以下语句为变量 ReceiveDate 和 ShipDate 创建了标签：

```
LABEL ReceiveDate = 'Date order was received'
      ShipDate = 'Date merchandise was shipped';
```

在 DATA 步中使用 LABEL 语句时，标签会成为数据集的一部分；但在 PROC 中使

用时，标签仅在该特定过程步中有效。

定制输出 你可以对过程步生成的输出进行很多控制。使用系统选项可以设置许多功能，例如居中、日期和页面方向（请参见第 1.13 节）。使用输出交付系统还可以更改输出的整体样式，产生不同格式（如 PDF 或 RTF）的输出，甚至更改输出的任何细节（第 5 章）。

输出数据集 大多数过程步可以生成某种报表，但有时你需要将过程步的结果保存为 SAS 数据集以便执行进一步分析。可以使用 ODS OUTPUT 语句（第 5.3 节）从任何过程输出创建 SAS 数据集。某些过程还可以使用 OUTPUT 语句或“OUT=” 选项将数据输出到 SAS 数据集。

4.2 使用 WHERE 语句在过程中生成子集

A	WW	WWWW
B	WW	WWWW
C	WW	WWWW
A	WW	WWWW
B	WW	WWWW

WHERE

A	WW	WWWW
A	WW	WWWW

任何读取 SAS 数据集的 PROC 步都有一个可选的 Where 语句。WHERE 语句告诉过程步使用数据的子集。你可能还记得有其他生成数据子集的方法，而可以选择不用 WHERE 语句。^① 但是 WHERE 语句方便快捷。取子集 IF 语句仅在 DATA 步中有效，而 WHERE 语句在 PROC 步和 DATA 步中都有效。

与在 DATA 步中生成子集不同，在过程步中使用 WHERE 语句不会创建新的数据集，这也是 WHERE 语句有时比其他生成子集的方法更加有效的一个原因。

WHERE 语句的基本形式：

WHERE *condition*;

只有满足语句中条件（*condition*）的观测才会被 PROC 使用。这可能看起来很熟悉，因为它类似于取子集 IF 语句。条件左侧是变量名，条件右侧是变量名、常数或数学表达式。数学表达式可以包含标准算术符号加号 (+)、减号 (-)、乘号 (*)、除号 (/) 和求幂 (**)。条件表达式两侧之间可以使用比较运算符和逻辑运算符，这些运算符可以是符号也可以是助记符。下面是一些常用的运算符：

^① 有关 WHERE 语句的详细信息，请参见附录。生成数据子集的其他方式包括取子集 IF 语句（第 2.13、3.7 和 6.9 节），以及“WHERE=” 选项（第 6.13 节）。

比较符	助记符	示例
=	EQ	WHERE Region = 'Spain';
$\emptyset =$, $\sim =$, $\wedge =$	NE	WHERE Region $\sim =$ 'Spain';
>	GT	WHERE Rainfall > 20;
<	LT	WHERE Rainfall < AvgRain;
\geq	GE	WHERE Rainfall \geq AvgRain + 5;
\leq	LE	WHERE Rainfall \leq AvgRain / 1.25;
&	AND	WHERE Rainfall > 20 AND Temp < 90;
, , !	OR	WHERE Rainfall > 20 OR Temp < 90;
	IS NOT MISSING	WHERE Region IS NOT MISSING;
	BETWEEN AND	WHERE Region BETWEEN 'Plain' AND 'Spain';
	CONTAINS	WHERE Region CONTAINS 'ain';
	IN (LIST)	WHERE Region IN ('Rain', 'Spain', 'Plain');

示例 你有一个包含知名画家信息的数据库。下面列出了其中的一部分数据。数据包含每位画家的姓名、主要风格和国籍：

Mary Cassatt	Impressionism	U
Paul Cezanne	Post-impressionism	F
Edgar Degas	Impressionism	F
Paul Gauguin	Post-impressionism	F
Claude Monet	Impressionism	F
Pierre Auguste Renoir	Impressionism	F
Vincent van Gogh	Post-impressionism	N

为了使该示例更加真实，它包含了两部分：一部分是创建永久 SAS 数据集，另一部分是生成数据子集。首先 DATA 步读取名为 Artists.dat 文件中的数据，然后使用直接引用（也可以用 LIBNAME 语句）的方法在 MySASLib 目录中（Windows）创建名为 STYLE 的永久 SAS 数据集。

```
DATA 'c:\MySASLib\style';
  INFILE 'c:\MyRawData\Artists.dat';
  INPUT Name $ 1-21 Genre $ 23-40 Origin $ 42;
RUN;
```

假设一天后你只想将印象派画家的列表打印出来。快速简便的方法是使用 WHERE 语句和 PROC PRINT。用引号起来的数据集名称告诉 SAS 这是一个永久 SAS 数据集。

```

PROC PRINT DATA = 'c:\MySASLib\style';
  WHERE Genre = 'Impressionism';
  TITLE 'Major Impressionist Painters';
  FOOTNOTE 'F = France N = Netherlands U = US';
RUN;

```

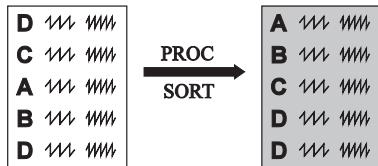
输出结果如表 4-1 所示。

表 4-1 Major Impressionist Painters

Obs	Name	Genre	Origin
1	Mary Cassatt	Impressionism	U
3	Edgar Degas	Impressionism	F
5	Claude Monet	Impressionism	F
6	Pierre Auguste Renoir	Impressionism	F

F = France; N = Netherlands; U = US

4.3 使用 PROC SORT 对数据排序



对数据排序有很多原因：需要为报表组织数据、在合并数据集之前、或者在另一个 PROC 或 DATA 步中使用 BY 语句之前。很幸运，PROC SORT 相当简单。该过程的基本形式为：

```

PROC SORT;
BY variable-list;

```

BY 语句中命名的变量称为 BY 变量。你可以指定任意数量的 BY 变量。在具有一个 BY 变量的情况下，SAS 根据该变量的值排序。在具有多个变量的情况下，SAS 先按第一个变量对观测进行排序，然后在第一个变量的类别内按第二个变量排序，以此类推。BY 组是 BY 变量取值相同的观测（多行数据）组成的一个组。例如，若 BY 变量是 State，则州名为 North Dakota 的所有观测会形成一个 BY 组。

控制输出数据集 “DATA=” 和 “OUT=” 选项指定输入数据集和输出数据集。若你没有指定 “DATA=” 选项，则 SAS 将使用最新创建的数据集。若你没有指定 “OUT=” 选项，则 SAS 将使用排序后的新数据集版本替换原始数据集。以下示例语句告诉 SAS 对名为 MESSY 的数据集进行排序，然后将排序后的数据输出到名为 NEAT 的数据集中：

```
PROC SORT DATA = messy OUT = neat;
```

“NODUPKEY” 选项告诉 SAS 删除具有相同 BY 变量值的观测。若你指定 “DUPOUT=” 选项，则 SAS 将删除的观测输出到指定的数据集中。要使用这些选项，只需将它们添加到 PROC SORT 语句中：

```
PROC SORT DATA = messy OUT = neat NODUPKEY DUPOUT = extraobs;
```

升序和降序排序 默认情况下，SAS 按升序（从最低到最高）对数据排序。要按降序对数据排序，在 BY 语句中每个需要降序排列的变量前，添加关键字 DESCENDING。以下语句告诉 SAS 先按 State 从 A 到 Z 排序，然后在 State 内按 City 从 Z 到 A 排序：

```
BY State DESCENDING City;
```

示例 以下数据显示了所选鲸鱼和鲨鱼的典型长度（英尺）。请注意，每一行包括多个物种的数据。

beluga	whale	15	dwarf	shark	.5	sperm	whale	60
basking	shark	30	humpback	.	50	whale	shark	40
gray	whale	50	blue	whale	100	killer	whale	30
mako	shark	12	whale	shark	40			

以下程序读取数据并进行排序：

```
DATA marine;
  INFILE 'c:\MyRawData\Lengths.dat';
  INPUT Name $ Family $ Length @@;
RUN;
/*对数据排序;
PROC SORT DATA = marine OUT = seasort NODUPKEY;
  BY Family DESCENDING Length;
PROC PRINT DATA = seasort;
  TITLE 'Whales and Sharks';
RUN;
```

DATA 步从名为 Lengths.dat 的文件中读取原始数据，然后创建名为 MARINE 的 SAS 数据集。之后 PROC SORT 按 Family 升序以及 Length 降序重新对观测进行排列。PROC SORT 的 “NODUPKEY” 选项删除 BY 变量值重复的观测，而 “OUT=” 选项将排序后的数据写入名为 SEASORT 的新数据集中。PROC PRINT 的输出如表 4-2 所示。

表 4-2 Whales 和 Sharks

Obs	Name	Family	Length
1	humpback		50.0
2	whale	shark	40.0
3	basking	shark	30.0
4	mako	shark	12.0
5	dwarf	shark	0.5
6	blue	whale	100.0
7	sperm	whale	60.0
8	gray	whale	50.0
9	killer	whale	30.0
10	beluga	whale	15.0

请注意，humpback 是第 1 个观测，对应的 Family 具有缺失值。这是因为对于数值变量和字符变量来说缺失值始终排序最低。此外“NODUPKEY”选项删除了 whale 鲸鱼的重复观测。日志中包含以下注释，显示出排序后的数据比原始数据集包含更少的观测。

NOTE: 从数据集 WORK.MARINE. 读取了 11 个观测

NOTE: 1 个具有重复键值的观测已删除。

NOTE: 数据集 WORK.SEASORT 有 10 个观测和 3 个变量

4.4 更改字符数据的排序顺序

乍一看字符数据排序很简单。毕竟大家都知道“A”在“B”的前面。但是“A”是否在“a”的前面就不那么明显了。SAS 提供了许多选项来控制字符数据的排序顺序（也称为排序序列）。本节将介绍其中几种。

ASCII 和 EBCDIC z/OS 操作环境的默认排序序列是 EBCDIC。其他大多数操作环境的默认排序序列是 ASCII。字符数据从最低到最高的基本排序顺序为：

ASCII	空格	数字	大写字母	小写字母
EBCDIC	空格	小写字母	大写字母	数字

若你仅在一种操作环境中工作，这对你来说无关紧要。但是若你需要在 Windows 上

创建将在 z/OS 上使用的数据集（反之亦然），则可能需要你的数据按照该操作系统所期望的顺序排序。你可以使用选项“SORTSEQ=EBCDIC”和“SORTSEQ=ASCII”更改排序顺序：^①

```
PROC SORT SORTSEQ = EBCDIC;
```

语义排序 默认情况下，大写字母和小写字母将分别排序，但是人们通常不这样排序。你可以使用语义排序生成更直观的顺序。“SORTSEQ=LINGUISTIC”选项带有“STRENGTH=PRIMARY”子选项，它告诉 SAS 忽略大小写。要使用这些选项，可以将它们添加到 PROC SORT 语句，如下所示：

```
PROC SORT SORTSEQ = LINGUISTIC (STRENGTH = PRIMARY);
```

下面的数据分别是未排序，按照默认 ASCII 顺序排序，以及按照忽略大小写的顺序排序：

未排序	默认排序	语义排序 (STRENGTH = PRIMARY)
Eva	ANNA	amanda
amanda	Zenobia	ANNA
Zenobia	amanda	eva
ANNA	eva	Zenobia

当对字符数据中的数字进行排序时，值“10”排在“2”前面。“NUMERIC_COLLATION=ON”子选项告诉 SAS 将数字等同于数值进行处理。

```
PROC SORT SORTSEQ = LINGUISTIC (NUMERIC_COLLATION = ON);
```

下面的数据分别是未排序、按照默认顺序排序、以及按照数值顺序排序：

未排序	默认排序	语义排序 (NUMERIC_COLLATION = ON)
1500m freestyle	100m backstroke	50m freestyle
200m breaststroke	1500m freestyle	100m backstroke
100m backstroke	200m breaststroke	200m breaststroke
50m freestyle	50m freestyle	1500m freestyle

^① “SORTSEQ=” 选项的其他可能值包括 DANISH、FINNISH、ITALIAN、NORWEGIAN、POLISH、SPANISH 和 SWEDISH。

示例 下面的数据包含一些姓名和地址。

```
Seiki    100 A St.          juneau    alaska
Wong     2 A St.           Honolulu  Hawaii
Shaw     10 A St. Apt. 10  Juneau    Alaska
Smith    10 A St. Apt. 2   honolulu hawaii
```

以下程序读取原始数据，然后对数据执行两次排序。数据先按 Street（使用数值顺序）排序，然后按 State（使用忽略大小写的顺序）排序。

```
DATA addresses;
  INFILE 'c:\MyRawData\Mail.dat';
  INPUT Name $6. Street $18. City $9. State $6. ;
RUN;
PROC SORT DATA = addresses OUT = sortone
  SORTSEQ = LINGUISTIC (NUMERIC_COLLATION = ON);
  BY Street;
PROC PRINT DATA = sortone;
  TITLE 'addresses Sorted by Street';
RUN;
PROC SORT DATA = addresses OUT = sorttwo
  SORTSEQ = LINGUISTIC (STRENGTH = PRIMARY);
  BY State;
PROC PRINT DATA = sorttwo;
  TITLE 'Addresses Sorted by State';
RUN;
```

结果如表 4-3、表 4-4 所示。

表 4-3 Addresses Sorted by Street

Obs	Name	Street	City	State
1	Wong	2 A St.	Honolulu	Hawaii
2	Smith	10 A St. Apt. 2	honolulu	hawaii
3	Shaw	10 A St. Apt. 10	Juneau	Alaska
4	Seiki	100 A St.	juneau	alaska

表 4-4 Addresses Sorted by State

Obs	Name	Street	City	State
1	Seiki	100 A St.	juneau	alaska
2	Shaw	10 A St. Apt. 10	Juneau	Alaska

(续表)

Obs	Name	Street	City	State
3	Wong	2 A St.	Honolulu	Hawaii
4	Smith	10 A St. Apt. 2	honolulu	hawaii

当你使用“STRENGTH=PRIMARY”选项时，将忽略 BY 组的大小写。在该示例中，值 alaska 和 Alaska 位于同一 BY 组中，hawaii 和 Hawaii 位于另一个 BY 组中。

4.5 使用 PROC PRINT 打印数据

PRINT 过程可能是使用最广泛的 SAS 过程了。本书中已多次出现使用该过程打印 SAS 数据集的内容。在它的最简单形式中，PROC PRINT 打印 SAS 数据集中所有变量的全部观测。SAS 会决定最佳输出格式，因此你不必担心一页上适合打印多少个变量这样的事情。但是还有一些其他的 PROC PRINT 的功能你也许也想用一用。

PRINT 过程仅需要一个语句：

```
PROC PRINT;
```

默认情况下，SAS 使用最新创建的 SAS 数据集。若你不想打印最新创建的数据集，则可以使用“DATA=” 选项指定数据集。建议你始终使用“DATA=” 选项在你的程序中明确指出所使用的数据集，因为快速识别出最后创建的数据集并不是件容易的事。

```
PROC PRINT DATA = data-set;
```

此外，SAS 会打印观测编号以及变量值。若你不需要观测编号，可以在 PROC PRINT 语句中使用 NOOBS 选项。若你使用 LABEL 定义变量标签，并且想打印标签而非变量名，也可以添加 LABEL 选项。以下语句显示了所有这些选项：

```
PROC PRINT DATA = data-set NOOBS LABEL;
```

下面是一些可能随手用得着的可选语句：

BY *variable-list*; BY 语句在输出中为每个 BY 变量值输出到新的区域，并在每个区域的最上方打印 BY 变量的值。数据必须按 BY 变量预先排序过。

ID *variable-list*; 当你使用 ID 语句时，不会打印观测编号。取而代之的是 ID 变量列表中的变量显示在页面左侧。

SUM variable-list; SUM 语句打印列表中变量总和。
VAR variable-list; VAR 语句指定打印的变量及其打印顺序。若不使用 VAR 语句，
 SAS 数据集中的所有变量会以它们在数据集中出现的顺序打印。

示例 四年级两个班级的学生通过卖糖果来赚取特殊实地考察的费用。挣钱较多的班级会获得一盒免费的糖果。下面是糖果销售结果的数据。依次显示学生姓名、班级号、交钱的日期、糖果类型（薄荷夹心巧克力为 MP、恐龙巧克力为 CD）以及售出的盒数：

Adriana	21	3/21/2012	MP	7
Nathan	14	3/21/2012	CD	19
Matthew	14	3/21/2012	CD	14
Claire	14	3/22/2012	CD	11
Ian	21	3/24/2012	MP	18
Chris	14	3/25/2012	CD	6
Anthony	21	3/25/2012	MP	13
Erika	21	3/25/2012	MP	17

每卖出一盒糖果，班级赚 1.25 美元。老师需要一份报表，给出每个班级挣到的钱数、每个学生挣到的钱数、售出的糖果类型以及学生上交钱的日期。以下程序读取数据，计算挣到的钱数 (Profit)，使用 PROC SORT 按班级对数据排序。然后，使用 PROC PRINT 的 BY 语句按 Class 打印数据，使用 SUM 语句给出总利润。VAR 语句列出要打印的变量：

```

DATA sales;
  INFILE 'c:\MyRawData\CandySales.dat';
  INPUT Name $ 1-11 Class @15 DateReturned MMDDYY10. CandyType $ 
        Quantity;
  Profit = Quantity * 1.25;
PROC SORT DATA = sales;
  BY Class;
PROC PRINT DATA = sales;
  BY Class;
  SUM Profit;
  VAR Name DateReturned CandyType Profit;
  TITLE 'Candy Sales for Field Trip by Class';
RUN;

```

表 4-5、表 4-6 是输出结果。请注意，变量 DateReturned 的值打印为 SAS 日期值。你可以使用下节介绍的输出格式以可读的形式打印日期。