

实战性强, 结合商业案例细致呈现SAS的优化建模方法, 深入讲解SAS数据处理、统计分析及时间序列, 涵盖引领大数据潮流的SAS高性能分析, 以及智能分析平台、解决方案、平台的安全性、高可用性等重要领域

深入解析SAS

数据处理、分析优化 与商业应用

夏坤庄 徐唯 潘红莲 林建伟/著



本书仅提供部分阅读，如需完整版，请联系QQ: 461573687

提供各种书籍pdf下载，如有需要，请联系 QQ: 461573687

PDF制作说明：

本人可以提供各种PDF电子书资料，计算机类，文学，艺术，设计，医学，理学，经济，金融，等等。质量都很清晰，而且每本100%都带书签和目录，方便读者阅读观看，只要您提供给我书的相关信息，一般我都能找到，如果您有需求，请联系我 QQ: 461573687, 或者 QQ: 2404062482。

本人已经帮助了上万人找到了他们需要的PDF，其实网上有很多PDF,大家如果在网上不到的话，可以联系我QQ。因PDF电子书都有版权，请不要随意传播，最近pdf也越来越难做了，希望大家尊重下个人劳动，谢谢！

备用QQ:2404062482

深入解析SAS

数据处理、分析优化
与商业应用

夏坤庄 徐唯 潘红莲 林建伟/著

BUSINESS
IMPLEMENTATION
STATISTICS
DATA MINING **SAS** OPERATIONS
ANALYTICS RESEARCH
FORECASTING PROGRAMMING



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

深入解析 SAS：数据处理、分析优化与商业应用 / 夏坤庄等著. —北京：机械工业出版社，2014.11

(数据分析技术丛书)

ISBN 978-7-111-48340-3

I. 深… II. 夏… III. 统计分析—应用软件 IV. C819

中国版本图书馆 CIP 数据核字 (2014) 第 246046 号

深入解析 SAS：数据处理、分析优化与商业应用

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）

责任编辑：陈佳媛

责任校对：殷虹

印刷：北京市荣盛彩色印刷有限公司

版次：2015 年 1 月第 1 版第 1 次印刷

开本：186mm×240mm 1/16

印张：55.25

书号：ISBN 978-7-111-48340-3

定价：99.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：(010) 88378991 88361066

投稿热线：(010) 88379604

购书热线：(010) 68326294 88379649 68995259

读者信箱：hzjsj@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问：北京大成律师事务所 韩光 / 邹晓东

华章IT
HZBOOKS | Information Technology



为什么要写这本书

数据和模型描述着世界，而 SAS 恰恰就是关于数据和模型的技术。SAS 技术在全球的数据处理和分析领域举足轻重。在国内，SAS 的应用日趋广泛，自然，对掌握 SAS 技术的人才需求也日益旺盛。

但是当大家谈及 SAS 的时候，普遍的一个感受是，掌握 SAS 比较难。这使我记起在 2000 年刚刚加入 SAS 中国公司几天后的一个下午，时任 SAS 中国区技术总监的栾世武博士问我：“怎么样？SAS 难学吗？”其实，在 SAS 公司的同事当中，大家并不会认为 SAS 有多难。究其原因，不过是如下几个：

- 在 SAS 公司，有着明确的路线图，大家可以清楚地知道学习 SAS 某个领域的顺序和步骤是什么。对于系统性非常强而且知识范围又较广的 SAS 而言，这是很重要的。
- 对于路线图中的每一个阶段，SAS 公司都提供了详尽的资料供阅读和学习。
- 有实际的项目去实践和锻炼。

上面所提到的因素，也正是大部分期望学习 SAS 技术的从业者快速有效掌握 SAS 的“窍门”。基于这样的经历和思考，几年以来我一直在构思这样一本书：

- 1) 以书中的章节结构来体现学习 SAS 核心内容的路线图。
- 2) 在每个章节的内容中，包含路线图中对应部分的必要学习资料，并且使得读者在读完相应的内容之后，有能力并且了解如何去学习更加深入和广泛的知识。
- 3) 提供贴近实际应用项目甚至有些复杂的例子，让读者领会解决实际问题的思路 and 技巧。

本书就是基于上述构思的一个实现，希望能够帮助大家系统地掌握 SAS 的专业知识，进

而从容地将其应用于商业实际中。

读者对象

本书主要适合于以下读者：

- 使用 SAS 进行数据抽取、转换和清洗的技术人员。
- 需要使用 SAS 对数据进行深入分析和数据挖掘的分析人员。
- 需要使用 SAS 进行时间序列预测和优化决策的建模专家。
- 使用 SAS 进行项目规划、实施和管理的系统架构师、系统管理员和项目管理人员。
- 团队的工作涉及 SAS 产品与技术的管理人员。

如何阅读本书

本书共 4 篇，系统介绍了 SAS 的核心技术模块和架构体系。

第一篇介绍 SAS 编程和数据处理（第 1 ~ 8 章）。内容包括如何运用 SAS 进行数据读入、处理和展现。掌握本书的这一篇内容可以满足大部分实际项目中数据处理的需要。该篇建议刚刚接触 SAS 的读者仔细研读，对 SAS 编程有全面了解的读者可以快速浏览或者在需要时查阅。

第二篇介绍 SAS 统计分析和时间序列预测（第 9 ~ 18 章）。内容既包括基本的理论介绍，又包括如何利用 SAS 去实现的具体技术。该篇建议需要学习数据分析、数据挖掘或进行预测的读者仔细阅读。

第三篇介绍 SAS 优化建模（第 19 ~ 24 章）。对于从事优化的读者来说，这一篇的内容将很有帮助。这一篇对常见的优化问题做了全面的介绍。其中的用例非常贴近现实，建议读者仔细研读。此外，建议从事优化的读者也学习一下第二篇中第 17 章关于时间序列分析的内容，因为在实际优化工作中，经常需要预测。

第四篇介绍 SAS 智能平台架构体系（第 25 ~ 28 章）。对于该篇内容，不需要像前 3 篇一样有深入的掌握，但这些内容对于项目规划和架构设计人员设计一个满足安全性、高可用性和高性能的 SAS 应用会非常有帮助。

致谢

本书的完成是整个写作团队合作的成果，蕴含着每一个作者的努力。

在本书即将完成之际，需要感谢的名字很多，把这长长的列表沉在深处之后，在此感谢我们所处的时代和我所在的 SAS 公司。时代赋予了企业和个人对数据进行分析 and 建模的需

求，SAS 公司给予了我们完成本书所需要的知识和使命感。

特别感谢机械工业出版社华章公司的 Lisa Yang。感谢 Lisa 的热情相邀和宝贵建议，促成了本书的完成，她的专业而高效的审阅，也使得本书增色极多。

夏坤庄 (Kansun Xia)

北京，2014 年 7 月

Preface

Why to write this book

The world can be described with data and model, and SAS is exactly about these two. SAS has been proved to be a prominent player in data management and advanced analytics field worldwide. In China, SAS is also getting widely used in more and more industries, as a result, talent demands of SAS expertise is growing as well.

However, when speaking of SAS, the first impression is that it is difficult to master. In an afternoon in 2000, not long after I joined SAS, Dr. Luan Shiwu, who was the CTO of SAS China, asked me, “ How do you feel about SAS? Is it difficult to learn?” Actually, like the majority of my colleagues, I do not think SAS is difficult to learn. Here are some reasons:

- ❑ SAS provides an explicit roadmap allowing the staff to learn specific domains of SAS in right order step by step. That is very helpful since SAS technology is highly systematically and broadly in scope of knowledge.
- ❑ Within each phase of the roadmap, elaborate materials are provided.
- ❑ Projects are available for practice and exercise.

I believe that the above reasons I mentioned are essential to anyone who wants to learn SAS. Basing on such experience and consideration, I have been conceiving of a book with features below:

- ❑ Implement the roadmap of learning SAS core knowledge by organizing chapters of this book.
- ❑ Provide essential learning materials corresponding to each phase of the roadmap in each chapter, so that readers are able to learn more deeply and widely after finishing each part.
- ❑ Provide examples that were extracted from business projects for readers so that they can learn approaches and skills of solving complicated business problems.

With this book, I hope it can help you master SAS technology systematically, further more you can apply it easily into your professional work.

Target audience

This book is mainly for the following people:

- Practitioners who use SAS to extract, transform and load data.
- Analysts who take advantage of SAS to do statistical analysis and data mining.
- Modeling experts who utilize SAS for time series forecasting and optimization.
- System architects, system administrators and project managers who use SAS to plan, implement, and manage projects.
- Managers whose team ' s work involves SAS product and technology.

How to read this book

This book consists of four parts, which systematically introduces SAS technology and architecture.

Part One (Chapter 1 - 8) is about SAS programming which covers how to use SAS to import, process and present data. After finishing this part, you can handle most of data processing work. It is suggested for new learners to read this part thoroughly, while for experienced users, you can quickly look through or refer to as it when needed.

Part Two (Chapter 9 - 18) talks about SAS statistical analysis and time series forecasting, including both fundamentals and specific technology for how to use SAS for implementation. People who need to learn data analysis, data mining or forecasting are recommended to read this part.

Part Three (Chapter 19 - 24) is about operational research. It is very useful for those who work on optimization modeling. This part gives fully description on common optimization problems with practical examples. In addition, people who work on optimization are suggested to learn Chapter 17 of Part Two, which is about time series forecasting, as forecasting usually goes along with optimization in actual work.

Part Four (Chapter 25 - 28) is about SAS platform architecture. This part is independent from the previous three parts. Project planners and architects will find this part very useful when they design a SAS application with high security, high availability, and high performance.

Contents

Part One (Chapter 1 - 8): SAS Programming and Data Processing

Chapter 1 Foundation of Base SAS

Chapter 2 Reading External Data to SAS Data Set

Chapter 3 SAS Data Set Processing

Chapter 4 Multiple Data Sets Processing

Chapter 5 Data Summary and Presentation

Chapter 6 SAS SQL Language

Chapter 7 SAS Macro Language

Chapter 8 SAS Programming with National Support

Part Two (Chapter 9 - 18): SAS Statistical Analysis and Time Series Forecasting

Chapter 9 Descriptive Statistical Analysis

Chapter 10 Parameter Estimation and Hypothesis Test

Chapter 11 Analysis of Variance

Chapter 12 Principal Component Analysis and Factor Analysis

Chapter 13 Cluster Analysis

Chapter 14 Discriminant Analysis

Chapter 15 Regression Analysis

Chapter 16 LOGISTIC Regression Analysis

Chapter 17 Time Series Analysis

Chapter 18 General process flow of SAS Data Mining

Part Three (Chapter 19 - 24): SAS Optimization Modeling

Chapter 19 Overview of Operational Research

Chapter 20 Fundamentals of Linear Programming

Chapter 21 Linear Programming with PROC OPTMODEL

Chapter 22 PROC OPTMODEL Programming

Chapter 23 Integer Linear Programming and Mixed Integer Linear Programming

Chapter 24 Examples of Optimization Modeling

Part Four (Chapter 25 - 28): SAS Business Implementation

Chapter 25 SAS Intelligence Platform and Solutions

Chapter 26 SAS Application Infrastructure Planning

Chapter 27 Security Administration of SAS Intelligence Platform

Chapter 28 High Availability of SAS Intelligence Platform

Acknowledgement

This book is the outcome of team work, including each writer's contributions.

With the completion of this book, there are too many people to whom I would like to express my appreciation. I would like to thank to this era and SAS. It is this era that endows organizations and individuals with the need of data analysis and modeling, and it is SAS that offers me the knowledge and the feeling of responsibility to complete this book.

In the end, I'm particularly grateful to Lisa Yang from Huazhang Company of China Machine Press for her warm invitation and precious suggestions which contribute to the completion of this book. Her professional and efficient review work improved this book a lot.

Kansun Xia
Beijing, China
July, 2014

目 录 *Contents*

前言

Preface

第一篇 SAS 编程和数据处理

第 1 章 Base SAS 基础 2

1.1 SAS 系统简介 2

1.2 启动 SAS 软件 4

1.2.1 SAS 窗口环境模式 4

1.2.2 非交互模式 5

1.2.3 批处理模式 6

1.2.4 交互式行模式 7

1.2.5 配置文件和 AUTOEXEC 文件 7

1.3 SAS 窗口环境 8

1.3.1 SAS 资源管理器 10

1.3.2 程序编辑器 10

1.3.3 日志 10

1.3.4 结果 11

1.3.5 输出 11

1.4 SAS 文件和逻辑库 11

1.5 一个简单的 SAS 程序 13

1.6 SAS Studio 17

1.7 本章小结 18

第 2 章 读取外部数据到 SAS 数据集 19

2.1 SAS 编程基本概念 20

2.1.1 SAS 逻辑库 20

2.1.2 SAS 数据集 23

2.1.3 SAS 逻辑库和数据集管理 31

2.1.4 SAS 系统选项 33

2.1.5 SAS 程序结构 37

2.2 通过 DATA 步读取数据 38

2.2.1 DATA 步处理 38

2.2.2 读取外部文本文件中的
数据 (初级) 42

2.2.3 读取外部文本文件中的
数据 (高级) 52

2.3 通过 IMPORT 过程读取外部
文件数据 62

2.4 访问关系型数据库系统中的数据 65

2.5 SAS 程序错误及处理 68

2.5.1 良好的 SAS 编程风格 68

2.5.2 常见错误及处理 69

2.6 本章小结 73

第3章 对单个数据集的处理	74	4.2 数据集的横向合并	130
3.1 选取部分变量	74	4.2.1 不使用 BY 语句实现横向合并	130
3.2 操作数据集的观测	78	4.2.2 使用 BY 语句实现横向合并	133
3.2.1 SAS 表达式	78	4.2.3 使用数据集选项 IN= 操作 观测	140
3.2.2 选取部分观测	84	4.3 数据集的更新	141
3.2.3 操作所选取的观测	88	4.4 数据集的更改	143
3.2.4 分组与排序	92	4.4.1 单个数据集的更改	143
3.3 创建新变量	96	4.4.2 两个数据集的更改	145
3.3.1 数据集选项 RENAME= 和 RENAME 语句	96	4.5 数据集处理的一点补充	146
3.3.2 赋值语句创建新变量	97	4.5.1 使用数据集选项 END=	146
3.3.3 对多个观测求和	99	4.5.2 使用自动变量 FIRST. 与 LAST.	148
3.4 循环和数组	103	4.5.3 使用 SET 语句中的选项 POINT= 和 NOBS=	149
3.4.1 循环	103	4.5.4 使用多个 SET 语句	150
3.4.2 SAS 数组	106	4.5.5 使用 HASH 对象处理多个 数据集	151
3.5 SAS 常用函数	109	4.6 本章小结	156
3.5.1 函数语法	109	第5章 数据汇总与展现	157
3.5.2 数值函数	110	5.1 通过 PRINT 过程制作报表	157
3.5.3 字符操作函数	110	5.1.1 制作简单报表	157
3.5.4 数值与字符转换函数	113	5.1.2 制作增强型报表	161
3.5.5 与日期时间相关的函数	115	5.1.3 改进报表显示	163
3.6 将数据集写出到外部文件	116	5.2 通过 TABULATE 过程制作汇总 报表	168
3.7 本章小结	119	5.2.1 制作基本汇总报表	168
第4章 对多个数据集的处理	120	5.2.2 制作高级汇总报表	172
4.1 数据集的纵向串接	120	5.2.3 改进报表显示	175
4.1.1 使用 SET 语句实现纵向串接	120	5.3 通过 GPLOT 过程制作图形	180
4.1.2 使用 APPEND 过程实现纵向 串接	127		
4.1.3 SET 语句与 APPEND 过程的 比较	129		

5.3.1	制作散点图	180	6.4.4	使用关键字 OUTER UNION 对表进行纵向合并	243
5.3.2	制作连线图	184	6.5	使用 SQL 管理表	245
5.3.3	制作多幅图形	188	6.5.1	使用 SQL 复制、创建与 删除表	245
5.3.4	制作气泡图	195	6.5.2	使用 SQL 插入行	247
5.4	通过 GCHART 过程制作图形	196	6.5.3	使用 SQL 删除部分行	248
5.4.1	制作柱状图	196	6.5.4	使用 SQL 修改表的列	249
5.4.2	制作分组柱状图	203	6.5.5	使用 SQL 更新列的值	250
5.4.3	制作饼图	206	6.6	本章小结	252
5.5	ODS 输出传送系统	210	第 7 章 SAS 宏语言		253
5.5.1	选择或删除输出对象	211	7.1	SAS 宏语言概述	253
5.5.2	创建多种格式输出文件	216	7.2	宏变量	254
5.6	本章小结	223	7.2.1	宏变量的定义	254
第 6 章 SAS SQL 语言		224	7.2.2	宏变量的调用	255
6.1	SQL 语言概述	224	7.2.3	宏变量的查看	257
6.2	使用 SQL 检索数据	225	7.2.4	宏变量的分类	258
6.2.1	SQL 的基本结构	225	7.2.5	宏变量的删除	259
6.2.2	使用 SQL 对列进行操作	226	7.3	宏函数	259
6.2.3	使用 SQL 对行进行操作	227	7.3.1	在宏语言中调用 SAS 函数	259
6.2.4	使用 SQL 对报表加工与 生成数据集	232	7.3.2	用宏函数处理算术与逻辑 表达式	260
6.2.5	子查询	233	7.3.3	常见的处理文本的宏函数	261
6.3	使用 SQL 对表进行横向合并	234	7.4	宏	263
6.3.1	使用 SQL 对表进行内连接	234	7.4.1	宏的定义与调用	263
6.3.2	使用 SQL 对表进行外连接	236	7.4.2	宏的存储	264
6.4	使用 SQL 对表进行纵向合并	237	7.4.3	宏的参数	266
6.4.1	使用关键字 EXCEPT 对表 进行纵向合并	238	7.4.4	宏与宏变量	269
6.4.2	使用关键字 INTERSECT 对表 进行纵向合并	240	7.5	宏语言与其他 SAS 语言	272
6.4.3	使用关键字 UNION 对表 进行纵向合并	241	7.5.1	宏语言的编译过程	272

7.5.2 宏语言与 DATA 步	274	9.2 描述性统计量	318
7.5.3 宏语言与 SQL 语言	277	9.2.1 描述数据集中趋势	319
7.6 宏编程	278	9.2.2 描述数据离散程度	320
7.6.1 条件语句	278	9.2.3 描述数据分布形态	322
7.6.2 循环语句	280	9.3 MEANS 过程的补充	334
7.7 本章小结	283	9.3.1 统计量列表	334
第 8 章 开发多语言支持的 SAS 程序	284	9.3.2 选项 WEIGHT= 和 WEIGHT 语句	335
8.1 多语言支持的基本概念	284	9.3.3 输出 SAS 数据集	336
8.1.1 语言 / 区域	285	9.3.4 WAYS 语句和 TYPES 语句	338
8.1.2 字符集和编码	285	9.4 本章小结	340
8.2 NLS 相关的 SAS 选项	288	第 10 章 参数估计与假设检验	341
8.2.1 语言 / 区域选项 LOCALE=	288	10.1 参数估计	341
8.2.2 编码选项 ENCODING=	290	10.1.1 点估计	341
8.2.3 时区选项 TIMEZONE=	295	10.1.2 区间估计	343
8.2.4 语言切换选项	296	10.2 假设检验	346
8.3 NL 格式和 NL 输入格式	297	10.2.1 基本原理	346
8.4 字符串和字符处理函数	302	10.2.2 T 分布与 T 检验	348
8.5 文本字符串外部化	303	10.2.3 TTEST 过程	350
8.6 本章小结	309	10.2.4 单样本均值 T 检验	352
第二篇 SAS 统计分析和时间序列预测		10.2.5 独立双样本均值 T 检验	354
第 9 章 描述性统计分析	312	10.2.6 配对样本均值 T 检验	360
9.1 基本概念	313	10.3 非参数假设检验	362
9.1.1 总体、个体和样本	313	10.4 分布拟合假设检验	365
9.1.2 简单随机抽样	313	10.5 本章小结	368
9.1.3 连续变量和分类变量	313	第 11 章 方差分析	370
9.1.4 参数、统计量和自由度	314	11.1 方差分析的基本原理	370
9.1.5 随机变量及概率分布	314	11.1.1 方差分析的模型	370
		11.1.2 方差分析的基本思想	371

11.1.3	方差分析的假设	373	12.2.3	使用 FACTOR 过程进行 主成分分析	396
11.2	单因素试验的方差分析	374	12.3	因子分析概述	399
11.2.1	TTEST 过程、ANOVA 过程与 GLM 过程的区别	374	12.3.1	公共因子与特殊因子	399
11.2.2	使用 ANOVA 过程进行 方差分析	374	12.3.2	因子分析的计算过程	400
11.2.3	使用 GLM 过程进行方差 分析	376	12.3.3	因子分析与主成分分析 比较	401
11.3	显著因素下的水平间差异检验	379	12.4	使用 SAS 实现因子分析	402
11.3.1	LSMEANS 语句与 MEANS 语句的区别	379	12.5	本章小结	407
11.3.2	利用 LSMEANS 语句进行 水平差异分析	380	第 13 章 聚类分析		408
11.4	双因素试验的方差分析	382	13.1	聚类分析的概述	408
11.4.1	双因素试验概述	382	13.1.1	聚类分析方法介绍与比较	408
11.4.2	利用 GLM 过程对不平衡 数据进行方差分析	383	13.1.2	相似性的度量	409
11.4.3	有交互作用因素的方差 分析	385	13.2	划分法与层次法	412
11.5	本章小结	386	13.2.1	使用过程 FASTCLUS 实现 K 均值聚类法	412
第 12 章 主成分分析与因子分析		387	13.2.2	使用过程 CLUSTER 实现 层次法	416
12.1	主成分分析概述	387	13.3	本章小结	422
12.1.1	主成分分析的基本思想	387	第 14 章 判别分析		423
12.1.2	主成分的定义、计算与 确定	389	14.1	判别分析概述	423
12.1.3	主成分分析难点探讨	390	14.1.1	判别分析的基本概念及 应用	423
12.2	使用 SAS 实现主成分分析	392	14.1.2	判别分析的假设条件	424
12.2.1	FACTOR 过程与 PRINCOMP 过程的比较	392	14.1.3	判别分析常见的方法	424
12.2.2	使用 PRINCOMP 过程进行 主成分分析	392	14.2	判别分析在 SAS 中的实现	426
			14.2.1	使用过程 DISCRIM 实现 一般判别分析	427
			14.2.2	使用过程 CANDISC 实现 典型判别分析	432

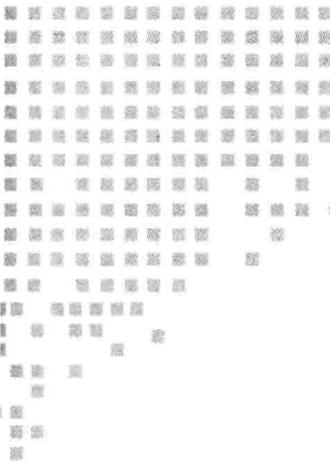
14.2.3	使用过程 STEPDISC 实现 逐步判别分析	436	16.3	LOGISTIC 过程的其他语句	482
14.3	本章小结	440	16.3.1	CLASS 语句	482
第 15 章	回归分析	441	16.3.2	ODDSRATIO 语句	483
15.1	变量关系探索	442	16.3.3	UNITS 语句	484
15.1.1	皮尔逊相关系数	442	16.4	建立模型	491
15.1.2	相关性检验	444	16.4.1	自变量与 Logit 值的关系	491
15.1.3	CORR 过程	444	16.4.2	自变量的互动作用	494
15.2	线性回归	448	16.4.3	模型选择	494
15.2.1	基本原理	449	16.5	本章小结	500
15.2.2	假设检验	451	第 17 章	时间序列分析	501
15.2.3	模型拟合	453	17.1	时间序列基本概念	501
15.2.4	模型选择	457	17.1.1	了解时间序列	501
15.2.5	模型预测	464	17.1.2	时间序列的数字特征	503
15.3	自变量间的共线性诊断	466	17.1.3	常见平稳和非平稳模型	506
15.4	本章小结	468	17.1.4	SAS 时间序列分析软件 简介	511
第 16 章	LOGISTIC 回归分析	470	17.2	平稳时间序列分析	511
16.1	基本原理	470	17.2.1	数据准备	512
16.1.1	线性概率模型	470	17.2.2	平稳性和白噪声检验	516
16.1.2	LOGISTIC 回归模型	471	17.2.3	模型识别	521
16.1.3	LOGISTIC 回归模型的 估计	473	17.2.4	参数估计和诊断检验	532
16.1.4	LOGISTIC 回归模型的 假设条件	474	17.2.5	预测	537
16.2	运用 LOGISTIC 过程拟合模型	475	17.3	趋势时间序列分析	546
16.2.1	基本语法	475	17.3.1	确定性时间趋势	546
16.2.2	假设检验	477	17.3.2	随机时间趋势	550
16.2.3	参数估计和解释	478	17.3.3	运用 ARIMA 过程建立 趋势模型	554
16.2.4	模型评价	480	17.3.4	异常点检测	564
			17.3.5	运用其他过程建立趋势 模型	565

17.4	季节时间序列模型	574	第 20 章 线性规划	632		
17.4.1	确定性季节因素	574	20.1	数学模型	632	
17.4.2	随机季节模型	578	20.1.1	问题的提出	632	
17.4.3	季节性诊断	579	20.1.2	线性规划问题	634	
17.5	本章小结	585	20.1.3	图解法	636	
第 18 章 SAS 数据挖掘的一般流程			586	20.2	单纯形法	638
18.1	SAS 数据挖掘概述	586	20.2.1	线性规划问题的标准型	638	
18.2	确定业务问题和数据准备	587	20.2.2	单纯形法的导出和运用	641	
18.2.1	确定业务问题	587	20.2.3	两阶段单纯形法	645	
18.2.2	数据准备	588	20.3	对偶理论和灵敏性分析	649	
18.3	数据抽样、探索与加工	590	20.3.1	对偶问题的导出	649	
18.3.1	数据抽样	590	20.3.2	对偶问题的基本性质	650	
18.3.2	数据探索	591	20.3.3	对偶单纯形法	651	
18.3.3	数据加工	601	20.3.4	对偶问题的经济解释	653	
18.4	数据建模	605	20.3.5	灵敏性分析	656	
18.4.1	模型的建立	605	20.4	内点法	657	
18.4.2	模型的评估	609	20.5	本章小结	658	
18.4.3	模型的实施	614	第 21 章 运用 PROC OPTMODEL			
18.5	本章小结	615	建立线性规划模型			
第三篇 SAS 优化建模			21.1	基本概念	659	
第 19 章 运筹学概述			618	21.1.1	参数	659
19.1	运筹学发展简介	618	21.1.2	索引和索引集	660	
19.2	优化模型的基本概念	621	21.1.3	数据类型	660	
19.3	优化模型的分类	623	21.1.4	名称	661	
19.4	优化建模步骤	624	21.1.5	表达式	661	
19.5	SAS/OR 简介	627	21.1.6	标识表达式	662	
19.6	一个简单的 OPTMODEL 程序	627	21.1.7	函数表达式	662	
19.7	本章小结	631	21.1.8	索引集的补充	662	
			21.2	基本结构	664	
			21.3	建立模型	667	

21.3.1	参数的声明	667	23.2	使用 PROC OPTMODEL 求解 混合整数线性规划	725
21.3.2	变量的声明	671	23.3	使用 0-1 变量建模	728
21.3.3	目标函数的声明	672	23.3.1	问题的提出	728
21.3.4	约束条件的声明	672	23.3.2	数学模型	728
21.3.5	求解器的调用	673	23.3.3	输入数据	731
21.3.6	数据输出	673	23.3.4	PROC OPTMODEL 代码和 输出	733
21.4	读取 SAS 数据集	678	23.4	本章小结	737
21.5	创建 SAS 数据集	686			
21.6	本章小结	688			
第 22 章 PROC OPTMODEL 程序设计			第 24 章 优化建模实例		
		689	24.1	集装箱问题	738
22.1	PROC OPTMODEL 中的流程 控制方法与集合运算	689	24.1.1	问题的提出	738
22.1.1	常见的流程控制方法	689	24.1.2	数学模型	739
22.1.2	常见的集合运算处理	695	24.1.3	输入数据	740
22.2	模型的更新	702	24.1.4	PROC OPTMODEL 代码和 输出	741
22.2.1	使用预求解器	702	24.1.5	功能与技巧汇总	747
22.2.2	决策变量的增加、固定 与限制	704	24.2	运输排程问题	748
22.2.3	约束的改变与放松	709	24.2.1	问题的提出	748
22.3	网络流模型	712	24.2.2	数学模型	751
22.3.1	网络流模型概述	712	24.2.3	输入数据	755
22.3.2	使用 OPTMODEL 求解 网络流模型	714	24.2.4	数据验证	759
22.4	本章小结	717	24.2.5	PROC OPTMODEL 代码和 输出	764
			24.2.6	功能与技巧汇总	773
			24.3	本章小结	773
第 23 章 整数线性规划和混合整数 线性规划			第四篇 SAS 智能平台架构体系		
23.1	整数线性规划和混合整数线性 规划概述	718	第 25 章 SAS 智能平台及行业解决 方案		
23.1.1	分支定界法	719			
23.1.2	割平面法	722	25.1	SAS 智能平台	777

25.1.1	数据层	778	26.2.1	SAS 应用的 I/O 特性	815
25.1.2	SAS 服务器	778	26.2.2	SAS 文件系统考虑	816
25.1.3	中间层	780	26.3	本章小结	818
25.1.4	客户端	780			
25.2	SAS 商业智能	781	第 27 章 SAS 智能平台安全管理	819	
25.2.1	SAS Office Analytics	781	27.1	身份标识	819
25.2.2	SAS Enterprise BI Server	784	27.1.1	用户	819
25.3	SAS 数据管理和集成	790	27.1.2	组	822
25.3.1	SAS 数据集成	791	27.1.3	角色	823
25.3.2	SAS 数据质量管理	792	27.2	认证	824
25.3.3	DataFlux 数据管理平台	792	27.2.1	认证机制	824
25.3.4	SAS 主数据管理	794	27.2.2	凭证管理	826
25.4	SAS 商业分析	796	27.2.3	认证到元数据服务器	832
25.4.1	SAS Enterprise Miner	796	27.2.4	认证到计算服务器	833
25.4.2	SAS Text Miner	798	27.2.5	认证到数据服务器	833
25.4.3	SAS 商业分析解决方案	799	27.2.6	单点登录	833
25.5	SAS 高性能分析	801	27.3	授权	834
25.5.1	SAS 内存分析	801	27.3.1	元数据授权	834
25.5.2	SAS In-Database	804	27.3.2	访问元数据文件夹	838
25.5.3	SAS 网格计算	804	27.3.3	访问数据	838
25.6	本章小结	805	27.3.4	访问 SAS 对象	839
			27.3.5	数据的细粒度控制	842
第 26 章 SAS 应用的架构规划		806	27.4	加密	843
26.1	SAS 应用的架构规划	806	27.4.1	加密提供方	843
26.1.1	SAS 应用的架构	806	27.4.2	加密 ODS PDF 文件	844
26.1.2	SAS Grid Manager 架构	811	27.4.3	SAS 加密系统选项	844
26.1.3	SAS 库内产品架构	812	27.4.4	PWENCODE 过程	844
26.1.4	SAS 内存分析产品架构	812	27.5	安全性审计	845
26.1.5	SAS 部署在高可用集群 中的架构	814	27.5.1	SAS 安全性报告宏	845
26.2	SAS 应用的 I/O 系统规划	815	27.5.2	SAS 日志模块	846
			27.5.3	Web 应用程序的日志	847

27.5.4 SAS 审计性能测量包	848	28.3.3 公共组件	857
27.6 本章小结	850	28.4 SAS 计算层	857
第 28 章 SAS 智能平台的高可用性	851	28.4.1 SAS 计算服务器负载均衡	857
28.1 高可用性相关概念	851	28.4.2 SAS 网格计算	858
28.2 SAS 高可用性方法概述	853	28.4.3 提高计算层组件可用性	858
28.2.1 高可用集群	853	28.4.4 作业运行选项	859
28.2.2 动态迁移	854	28.5 SAS 中间层	860
28.2.3 SAS 环境备份和恢复	854	28.5.1 SAS Web Application Server 集群	860
28.3 SAS 元数据服务器	855	28.5.2 提高中间层组件的可用性	861
28.3.1 元数据服务器集群	856	28.6 数据层	862
28.3.2 提高元数据服务器可用性	856	28.7 本章小结	863



第一篇 *Part 1*

SAS 编程和数据处理

- 第 1 章 Base SAS 基础
 - 第 2 章 读取外部数据到 SAS 数据集
 - 第 3 章 对单个数据集的处理
 - 第 4 章 对多个数据集的处理
 - 第 5 章 数据汇总与展现
 - 第 6 章 SAS SQL 语言
 - 第 7 章 SAS 宏语言
 - 第 8 章 开发多语言支持的 SAS 程序
- 



Chapter 1

第 1 章

Base SAS 基础

本章将从 SAS 系统开始，介绍 Base SAS 的组成部分，并以 Windows 环境为例介绍 SAS 窗口环境、SAS 逻辑库、数据集、目录 (Catalog) 等 SAS 中常用的概念。在了解了这些基础知识之后，会引导读者使用以上的知识编写一段简单代码，提交执行，并查看日志及运行结果。最后将用简短的篇幅简单介绍 SAS 最新推出但将会承担重要角色的 SAS Studio 的基本功能。

需要注意的是，本书中描述的内容会包括 Windows 和 UNIX (和 Linux) 操作系统，如果在 Windows 和 UNIX 环境下的操作或命令有所不同，将会专门说明。本书内容未专门考虑 Mainframe，因为其操作使用模式相差很多，而且读者会较少接触和使用 Mainframe 环境，但书中对 SAS 软件和产品的描述、编程概念和程序语言以及给出的代码在 Mainframe 环境下同样适用。关于 SAS 的版本，本书是基于写作时发布的最新版 SAS 9.4 来展开的，除非特别说明，书中内容也同样适用于较早的版本 SAS 9.3 和 SAS 9.2。

本章对 Base SAS 窗口环境进行了着重介绍，目的在于让读者学会如何使用 SAS 窗口环境开发、运行 SAS 代码，并查看结果和检查代码运行日志。但是书中不会介绍每个菜单、子菜单、工具栏以及其他在 Base SAS 软件中出现的元素和功能，因为读者在实际学习和工作中可以很方便地通过 SAS 软件提供的帮助文件进行了解。

1.1 SAS 系统简介

SAS 提供了一套集成的可扩展的解决方案和使用灵活、功能强大的 SAS 编程语言，用于执行如下任务：数据输入和获取、数据转换处理和管理、报表绘制和图形、统计和数学分析、商业规划、预测、运筹优化，以及应用开发等。

SAS 可以在多种操作系统下运行，包括 Windows、UNIX、Linux 以及 Mainframe 等。同时，SAS 程序代码具有很好的移植性，在一种环境下开发的 SAS 代码可以在其他操作系统下运行。

SAS 系统的核心 Base SAS 由以下部分组成。

- DATA 步：用于处理和管理数据。
- SAS 过程 (Procedure)：用于分析、处理和制作报表。
- 可扩展和定制 SAS 软件程序的宏语言 (Macro Facility)：可以减少程序文本，使 SAS 程序编写得更有效且易于维护，便于编写更为复杂的程序逻辑。
- DATA 步调试器：当提交的 DATA 步运行出错或产生的输出结果与预期不一致时，可以借助它来跟踪 DATA 步的执行情况，从而帮助发现程序逻辑中的错误。
- 输出交付系统 (Output Delivery System, ODS)：该系统会产生各种易于访问的格式输出，例如，HTML 文件、传统的列表输出、PostScript 文件、RTF 文件和输出数据集等。
- SAS 窗口环境：它是一个开发和测试 SAS 程序的交互式图形用户界面，本节后面会有更进一步的介绍。

这其中，前面 3 个是 SAS 语言的主要元素，本篇后面的章节会专门介绍。

Base SAS 软件提供数据处理过程和基础的统计过程 FREQ、MEAN、CORR 及 UNIVARIATE 等，可以与其他的 SAS 产品一起使用，从而实现更强大的数据读取、分析、优化、展示等功能。下面列出了部分常用的 SAS 产品，用于实现数据读取、统计分析、优化和信息展示等功能。

(1) SAS/ACCESS 接口

提供与各种第三方数据源进行交互的功能。例如各种关系型数据库，诸如 Oracle、DB2、Teradata 等；ERP 系统诸如 SAP R/3、PeopleSoft 等；同样对于 Hadoop 等也有专门的 ACCESS 接口。对于不同的数据源，ACCESS 接口需要单独的软件使用许可。SAS 与第三方的数据源进行交互时，将直接调用该数据库或应用厂商提供的客户端对数据进行访问，从而保证了与数据访问的效率。此外，SAS/ACCESS 还提供接口访问 Microsoft Access 数据库文件和 Excel 工作簿文件中的数据。

(2) SAS/GRAPH

SAS/GRAPH 是 SAS 系统的数据可视化和展现 (图形) 组件，用于数据和信息展现，并且它可通过二维和三维图形 (包括图表、散点图和地图)，可视化地展现数据值之间的关系。还可创建文本幻灯片、生成各种图形输出，并可提供实用程序和管理输出。

(3) SAS/STAT

SAS/STAT 软件提供了全面的统计分析方法，共有超过 75 个统计分析过程，包括 T 检验、方差分析 (ANOVA 过程)、聚类分析 (CLUSTER 过程、VARCLUSTER 和 FASTCLUS 过程)、因子分析 (FACTOR 过程)、回归分析 (REG 过程)、逻辑斯蒂 (LOGISTIC 过程) 等。

SAS/STAT 软件还包括效能和样品容量分析 (PSS) 应用程序。该软件不断被更新, 以反映新的研究成果和方法。

(4) SAS/ETS

提供用于经济计量分析、时间序列分析和预测 (ESM 过程、ARIMA 过程和 UCM 过程等)、系统建模与仿真 (MODEL 过程)、离散选择分析、定性有限因变量模型分析、时间序列数据的季节性调整、财务分析和报告、访问经济和金融数据库及时间序列数据的管理。除了以上过程外, SAS/ETS 软件还包括对经济和金融数据库以及互动环境的无缝访问, 从而进行时间序列预测及投资分析。

(5) SAS/OR

SAS/OR 专注于运筹与优化。SAS/OR 提供的 OPTMODEL 建模语言用于构建、解决和维护最优化模型的建模环境, 通过 OPTMODEL 过程的各种求解器或单个过程, 例如 OPTLP、OPTMILP、OPTMILP 过程, 解决线性规划、混合整数规划、非线性规划等问题。

以 Base SAS 软件和以上产品与技术作为基础, 构建在 SAS 智能平台 (SAS Intelligence Platform) 上的 SAS 许多商业解决方案, 可以帮助各类商业客户和其他组织机构解决诸多业务领域的特定问题, 例如客户智能、风险管理、供应链、零售等。关于 SAS 商业解决方案的内容, 在本书的第四篇会有相应的介绍。

1.2 启动 SAS 软件

SAS 有多种运行模式: SAS 窗口环境模式、非交互式模式、批处理模式及交互式行模式, 下面会一一介绍。除了上面提到的 4 种模式外, SAS 还可运行在对象服务器模式里, SAS 元数据服务器、工作区服务器、存储过程服务器和 OLAP 服务器都是属于这种模式。关于这些服务器, 在本书第四篇会进行讨论。

1.2.1 SAS 窗口环境模式

SAS 窗口环境是 SAS 提供了一种交互式图形界面, 是在 Windows 环境下使用 SAS 编辑或提交 SAS 程序语句最方便也是最常用的模式。在 SAS 窗口环境中, 用户可以通过程序编辑器编辑并提交 SAS 语句, 程序语句的执行状态、执行时间等日志信息及 put 语句的输出会显示在日志窗口, 同时还会提供在线帮助等。本章下一节会使用 Windows 环境下的窗口环境作为示例, 详细介绍 SAS 窗口环境的各个窗口功能及其使用。

在 Windows 环境下启动 SAS 窗口环境和启动其他 Windows 应用程序一样有多种方式, 可通过“开始”菜单里的快捷方式、命令行等方式进行。在安装 SAS 软件时, SAS 软件安装程序会提示选择要安装的 SAS 语言版本。如果当前操作环境下安装了多种语言的 SAS, 英文的 SAS 可以通过“开始”→“程序”→SAS→SAS 9.4 (English) 启动。启动所有语言 (包括英文) 的 SAS 软件时, 其快捷方式位于“开始”→“程序”→SAS→Additional Languages 中。例如, 启动 Windows 操作环境下简体中文 SAS 软件的快捷方式为: “开始”→“程序”→

SAS → Additional Languages SAS 9.4 (Chinese (Simplified)), 如图 1.1 所示。

此外, 还可以使用命令行方式启动 SAS 窗口环境。在下面给出的 Windows 和 UNIX 操作环境下的命令后, 都可以指定其他系统选项来定制要启动的 SAS 会话。例如, 选项 `-NODATE` 表示在该 SAS 会话中产生的输出页面里不显示日期, 选项 `-CONFIG` 指定 SAS 配置文件, 以在启动时加载配置文件中更多的系统选项等。

□ Windows 环境

```
C:\>"C:\Program Files\SASHome\SASFoundation\9.4\sas.exe"
```

□ UNIX 环境

```
#!/opt/SASHome/SASFoundation/9.4/sas -dms
```

UNIX 环境下的命令行若不加选项 `-DMS`, 则会进入 SAS 的显示管理系统。当使用 Windows 机器通过 Telnet 远程登录 SAS 软件所在的 UNIX 主机时, 如果需要使用 SAS 窗口环境, 可以在该 Windows 机器上启动 X-Windows 软件, 例如 Exceed、Xming、Cygwin 等, 并设置当前 Telnet 会话的 `DISPLAY` 环境变量到该 Windows 机器上。这样, 所启动的 SAS 窗口环境会重定向到该 Windows 操作系统。当启动 SAS 的显示管理系统时, 在该 Windows 环境下会弹出类似的 SAS 窗口环境。在初次使用 X-Windows 窗口时会有些不习惯, 有些操作与 Windows 环境下的 SAS 窗口稍有差异, 但大部分都很类似。

在 UNIX 环境下, 更多使用的是非交互模式或批处理模式, 或者其他的工具。例如, 可使用 Windows 环境下的客户端程序 SAS Enterprise Guide 将 SAS 代码提交到 UNIX 服务器上。

1.2.2 非交互模式

很多情况下, SAS 程序保存在外部操作系统文件中, 这时可以在不启动 SAS 窗口环境的情况下非交互式地提交该文件。使用非交互模式执行保存在外部文件中的 SAS 程序语句时, SAS 会打开该文件, 执行该文件中的程序, 并将日志和输出根据操作环境写入指定文件中。当该文件中的 SAS 程序执行完成时, SAS 自动退出。下面是非交互模式下使用 SAS 执行外部文件中的 SAS 程序语句示例。

□ 在 Windows 环境下的示例如下:

```
C:\>"C:\Program Files\SASHome\SASFoundation\9.4\sas.exe" -sysin
C:\sas\code\test.sas -log C:\sas\logs\test.log -print
C:\sas\lst\test.lst
```

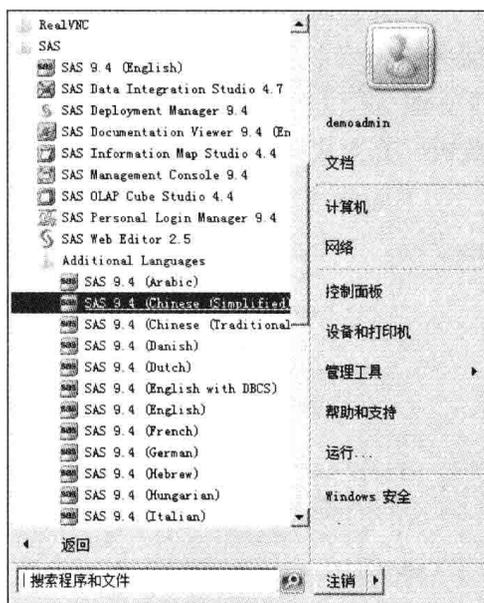


图 1.1 启动简体中文 SAS 窗口环境

□ 在 UNIX 环境下的示例如下：

```
#/opt/SASHome/SASFoundation/9.4/sas -sysin /opt/sas/code/test.sas -log /opt/sas
  /logs/test.log -print /opt/sas/lst/test.lst
```

选项“-sysin”指定 SAS 程序语句所在的文件。当要执行的 SAS 程序语句文件紧接着 sas 命令时，该选项可省略。“-log”指定输出日志文件，“-print”指定执行输出文件。

1.2.3 批处理模式

在支持批处理或后台执行的操作环境下，SAS 还可运行在批处理模式中。用户可以将上面的一个或多个非交互模式中的命令写入批处理脚本，并保存在批处理 .bat（Windows 环境下）或 .sh（UNIX 环境）文件中，然后提交该批处理文件执行。当以批处理模式提交 SAS 作业时，会生成两个文件，它们分别包含该作业执行的 SAS 日志和输出。

下面给出 UNIX 下的 .sh 文件内容，该批处理文件提交了两个 SAS 程序执行 test.sas 和 test2.sas。Windows 环境下类似，但必须使用相应的脚本语法。

```
#!/bin/sh
cd /opt/sas
/opt/SASHome/SASFoundation/9.4/sas /opt/sas/code/test.sas -log /opt/sas
  /logs/test.log -print /opt/sas/lst/test.lst &
wait
Sleep 15
/opt/SASHome/SASFoundation/9.4/sas /opt/sas/code/test2.sas -log /opt/sas
  /logs/test2.log -print /opt/sas/lst/test2.lst &
wait
```

当使用调度软件或操作系统调度命令对 SAS 作业进行预定执行时，通常使用该模式。这样可以使执行时间较长的 SAS 作业在晚间或其他预定时间执行，或根据业务需要定期自动执行。下面给出了在 Windows 和 UNIX 环境下使用操作系统计划或预定功能批处理执行的示例。在 SAS 商业智能解决方案中这种模式也经常使用，SAS 智能平台也提供了与第三方调度软件和操作系统调度服务的集成。

1. Windows 环境

使用 Windows 提供的任务计划程序，依次选择“开始”→“所有程序”→“附件”→“系统工具”→“任务计划程序”，并指定任务要执行的操作脚本为上面提到的批处理文件。如图 1.2 所示为在 Windows“任务计划程序”中预定为在每天 00:30:00 执行的 SAS 作业，所设定操作的启动程序为包含 SAS 命令的 .bat 文件。

2. UNIX 环境

使用 crontab 命令预定上面提到的批处理文件。下面的 crontab 文件内容给出了预定在每天凌晨 00:30:00 执行批处理文件 sasjob.sh 中的作业。

```
30 00 * * * /bin/ksh /opt/sas/scheduler/sasjob.sh > /opt/sas/scheduler/sasjob.log 2>&1
```

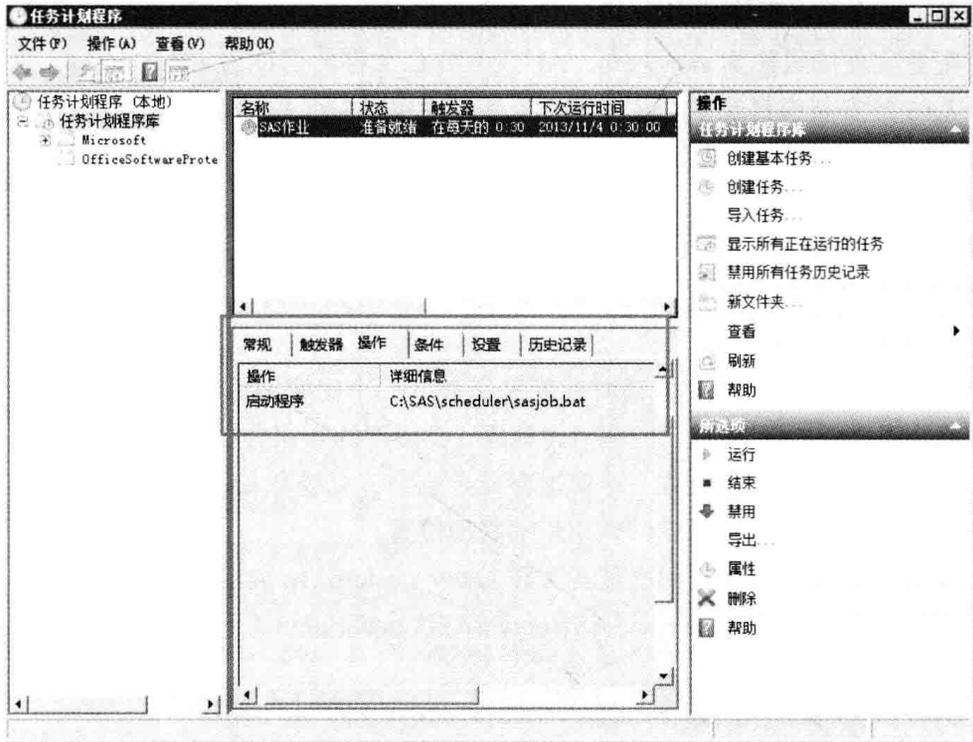


图 1.2 Windows 环境下的“任务计划程序”窗口

1.2.4 交互式行模式

该模式在 UNIX 操作系统中可用，是一种较少使用的模式。但作为 SAS 支持的启动模式，在这里也简单地介绍一下。在交互行模式下，顺序地输入程序语句，所输入的 DATA 步或 PROC 步当遇到 RUN、QUIT、分号、另一个 DATA 步或 PROC 步，或者 ENDSAS 语句时会提交执行。同时随着 DATA 步或 PROC 步的提交，这些程序语句的日志和输出（如果有输出的）也会立即显示。可使用 NODMS 或 NODMSEXP 系统选项启动交互式行命令模式的 SAS 会话。使用 NODMS 选项的命令示例如下：

```
#/opt/SASHome/SASFoundation/9.4/sas -nodms
```

后面还可以接其他系统选项或使用 -CONFIG 选项指定 SAS 配置文件。按 EOF 键 (Ctrl+D 组合键) 或提交 ENDSAS 语句将结束交互式行命令模式的 SAS 会话。示例如下：

```
5? endsas;
```

1.2.5 配置文件和 AUTOEXEC 文件

使用上述任何一种模式启动 SAS 时，都可以通过定义配置文件和 AUTOEXEC 文件来定制 SAS 会话。在这两个文件中可以指定系统选项和任何时候启动 SAS 会话时自动执行的 SAS 语句。SAS 系统选项控制 SAS 会话的许多方面，包括输出目的地、程序执行效率及

SAS 文件和逻辑库的属性等。

SAS 配置文件的名称为 `sasv9.cfg`，AUTOEXEC 文件的名称为 `autoexec.sas`，这些文件通常都位于 SAS 的安装根目录下。在多语言安装环境中，SAS 根目录下的 `nls` 目录中还包括各种语言适用的 SAS 配置文件所在目录，例如 `en` 是英文、`zh` 是简体中文、`zt` 是繁体中文。启动 SAS 时如果不指定配置文件，在 Windows 操作环境下，SAS 会去查找 SAS 安装根目录下的 `sasv9.cfg` 文件。通过 Windows “开始” 菜单启动各种语言的 SAS 软件时，SAS 会自动调用对应语言的配置文件。而在 UNIX 操作环境中，SAS 启动时所使用的配置文件是在启动 SAS 的文件中指定的。

有时，也会需要定制 SAS 启动时的配置文件。一个比较实际的例子就是，在很多商业项目中，不希望 SAS 使用其默认的临时逻辑库（逻辑库的知识会在后面介绍）`WORK`，而是希望将临时逻辑库 `WORK` 建立在一个高速存储系统中，以提高 SAS 对某些常用操作的效率，这时就可以在配置文件中指定临时 `WORK` 的物理位置。

在 Windows 环境下使用定制的配置文件 `sasv9_custom.cfg` 的示例如下。在该示例中，假定该文件位于目录 `C:\Program Files\SASHome\SASFoundation\9.4` 下（在 UNIX 环境下类似）。

```
C:\> "C:\Program Files\SASHome\SASFoundation\9.4\sas.exe" -config
"C:\Program Files\SASHome\SASFoundation\9.4\sasv9_custom.cfg"
```

系统选项 `AUTOEXEC` 用于指定 `AUTOEXEC` 文件。`AUTOEXEC` 文件包含启动 SAS 或其他 SAS 进程时自动执行的 SAS 语句，例如该文件可包含一些定义在 SAS 会话中的经常使用的 SAS 逻辑库的 `LIBNAME` 语句。

SAS 启动时，如果没有指定 `AUTOEXEC` 或 `NOAUTOEXEC` 选项，SAS 会在当前目录、用户目录和 SAS 安装根目录下查找 `AUTOEXEC` 文件。可以在启动 SAS 的命令行里指定 `AUTOEXEC` 选项，还可以通过 `SASV9_OPTIONS` 环境变量将该选项放入配置文件里。

在 Windows 环境下通过命令行指定 `autoexec` 文件 `autoexec_custom.sas` 的示例如下。在该示例中，假定该文件位于目录 `C:\Program Files\SASHome\SASFoundation\9.4` 下（在 UNIX 环境下类似）。

```
C:\> "C:\Program Files\SASHome\SASFoundation\9.4\sas.exe" -AUTOEXEC
"C:\Program Files\SASHome\SASFoundation\9.4\autoexec_custom.sas"
```

1.3 SAS 窗口环境

SAS 窗口环境是一个开发、调试和运行 SAS 程序的交互式图形用户界面。通过 SAS 窗口环境，用户可以交互式地编辑和执行 SAS 代码、显示 SAS 日志、查看 SAS 过程的输出以及在线帮助，同时还可以通过图形界面操作数据和改变 SAS 系统设置。SAS 窗口环境通常在 Windows 系统下使用，所以本书后面的章节都将以 Windows 环境下的 SAS 窗口环境进行说明。

SAS 软件启动后的界面包括菜单、命令框、工具栏、窗口、窗口条以及状态栏，同时还支持浮动菜单、Windows 环境快捷键（比如粘贴快捷键 Ctrl+C 及剪贴板功能），以及 Base SAS 软件本身提供的快捷键（比如，提交代码执行“F3”）。

SAS 菜单包括在当前上下文环境下可选择的选项列表，当正在使用的窗口发生变化时，菜单项会随之发生变化。例如，如果当前窗口是“资源管理器”，那么菜单视图会显示在“资源管理器”窗口可用的视图选项。如果“程序编辑器”是当前窗口，那么菜单视图会显示在“程序编辑器”窗口可用的视图选项。“工具栏”则显示为窗口按钮或图标。当单击“工具栏”里的工具项时，会产生对应的功能或动作。例如，单击“工具栏”里打印机的图标会开始打印过程。同样工具栏中的可选工具项也和当前的活动窗口相关。命令框位于工具栏左侧。可以在命令框输入命令行，例如打开 SAS 窗口和获取帮助信息。

下面来介绍 SAS 窗口环境的 6 个主要窗口：“程序编辑器”、“日志”、“输出”、“结果”、“SAS 资源管理器”和“编辑器”。第一次启动时，默认打开的窗口为“程序编辑器”、“日志”、“输出”和“SAS 资源管理器”窗口，“输出”窗口隐藏在其他窗口后面。所打开的窗口和窗口布局与 SAS 所在的操作环境相关，例如，在 Windows 环境下，“增强型编辑器”会代替“程序编辑器”。如图 1.3 所示为 Windows 环境下 Base SAS 软件的窗口环境，其中“结果”窗口和“SAS 资源管理器”共用窗口，可通过窗口下端的选项卡进行切换。提交 SAS 程序执行完成后，默认的 HTML 输出会展示在“结果”窗口中。

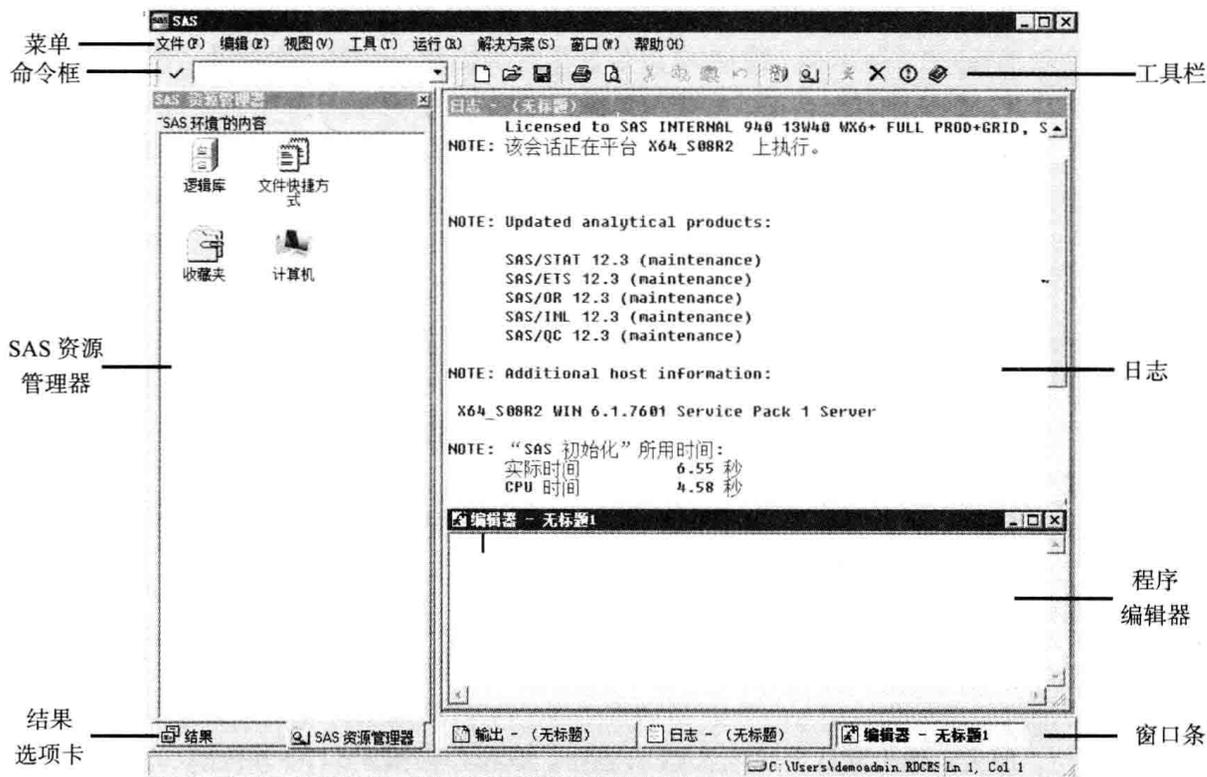


图 1.3 Windows 环境下的 SAS 窗口环境

在任一时间，将只有一个窗口处于激活状态，该窗口称为当前窗口或活动窗口，可以通过单击“窗口条”的窗口标签激活对应窗口。在命令栏输入相应命令并按回车键，或在菜单视图的子菜单中也可以打开并激活对应的窗口。

1.3.1 SAS 资源管理器

“SAS 资源管理器” (Explorer) 窗口用于管理该窗口环境中的文件，包括查看 SAS 文件列表、创建新的 SAS 文件，查看、添加或删除逻辑库，创建外部文件的快捷方式，移动、复制和删除文件，打开相关的窗口（比如新建逻辑库窗口）等。该窗口最常用的功能是管理逻辑库及逻辑库中的 SAS 文件，相关内容将在 1.4 节介绍。

“SAS 资源管理器”以树状结构管理当前 SAS 环境中的文件，最上层显示的图标为“逻辑库”、“文件快捷方式”、“收藏夹”和“计算机”，如图 1.3 所示。可以通过双击每个图标进入其下层的内容或打开一个文件。如果当前不在最上层，可以通过菜单“视图”→“向上一级”，或工具栏上的工具项，返回至上一级。还可通过菜单“视图”→“显示树状结构”打开两级窗口。

“SAS 资源管理器”窗口可通过在命令框中输入 EXPOLORE 并按回车键来打开，或者选择菜单“视图”→“SAS 资源管理器”打开。

1.3.2 程序编辑器

“程序编辑器” (Program Editor) 窗口用于输入、编辑、提交和保存 SAS 程序。该窗口还可通过在命令框输入 PROGRAM 或 PGM 并按回车键来打开，或者选择菜单“视图”→“程序编辑器”打开。默认设置下，在“程序编辑器”窗口中，代码提交后就会被清除，可在命令框中输入 recall 重新显示。可通过修改如下设置来取消清除代码：在当“程序编辑器”窗口为活动窗口时，选择菜单“工具”→“选项”→“程序编辑器”打开的“程序编辑器选项”对话框的“编辑”选项卡，取消勾选“提交时清除文本”选项。此外，在该对话框中还可以修改其他设置。

在 Windows 环境下，默认打开的“程序编辑器”窗口为“增强型编辑器”。“增强型编辑器”具有更加丰富的功能，可折叠和展开代码段，还可以通过设置使其在窗口左侧边缘显示行号。可通过在命令框输入 wnextedit 或 wpgm 并按回车键来打开或切换“增强型编辑器”，或者选择菜单“视图”→“增强型编辑器”打开。

在编辑器中输入的 SAS 程序代码可保存到文件系统的文件中。

1.3.3 日志

“日志” (Log) 窗口可查看当前 SAS 会话和 SAS 程序的消息。如果提交的程序产生意外结果，日志消息会提示错误信息，可以帮助找出 SAS 程序或设置的错误。如果 SAS 程序中有 PUT 语句，那么该输出默认会写到 SAS 日志中。在命令框输入 LOG 并按回车键，或选择菜单“视图”→“日志”，可打开“日志”窗口。日志同样也可保存到文件系统中，以便于以

后查看。作为 SAS 开发或使用人员，要养成每次代码提交执行完成后首先检查 SAS 日志的习惯。

1.3.4 结果

通过“结果”（Results）窗口可查看在该窗口环境提交的 SAS 代码的输出列表。在 SAS 9.4 中，默认输出为 HTML 格式。“结果”窗口以树形结构列出 SAS 程序代码执行后产生的输出。提交 SAS 代码后，HTML 内容显示在“结果浏览器”窗口，文件名称展示在“结果”窗口中。可以查看、保存或打印单个结果文件。

在命令栏输入 ODSRESULTS 并按回车键，或者选择菜单“视图”→“结果”，可打开“结果”窗口。

1.3.5 输出

可通过“输出”（Output）窗口查看 SAS 程序的列表（LISTING）输出。默认情况下，“输出”窗口位于其他窗口后面。当 SAS 程序产生了列表输出时，“输出”窗口会自动移动到显示前面。可在命令栏输入 OUTPUT、OUT、LISTING 或 LST 并按回车键来打开“输出”窗口，或者选择菜单“视图”→“输出”打开。

从 SAS 9.3 开始，SAS 的默认输出从列表输出变成了 HTML。可以通过 ODS 语句打开列表输出，产生列表输出的同时也会生成 HTML，不再需要列表输出时可再使用相应的 ODS 语句关闭该类型输出。还可使用菜单“工具”→“选项”→“参数选择”对话框的“结果”选项卡，选择输出类型和设置系统参数，参数选择对话框的默认设置如图 1.4 所示。勾选“创建列表”复选框会打开 SAS 软件的列表输出，还可选择 HTML 的样式，默认为 HTMLBlue。

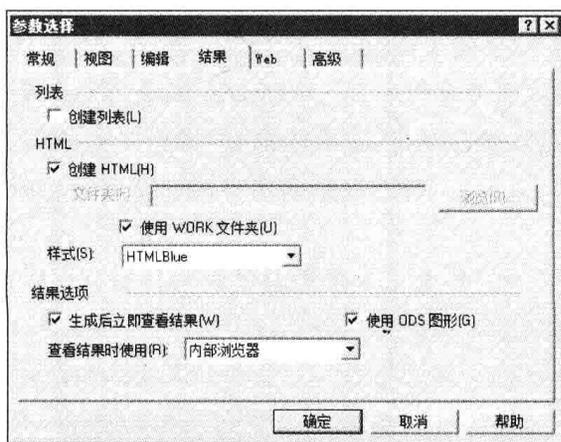


图 1.4 结果参数设置

1.4 SAS 文件和逻辑库

在熟悉了 SAS 窗口环境后，接下来了解一下 SAS 文件和 SAS 管理文件的方式。SAS 文件是指由 SAS 创建、维护和管理，并且 SAS 知道其结构的文件，例如 SAS 数据集、目录（Catalog）等。通常这些文件也表现为操作环境中的文件，操作环境也会对它们进行管理。所有的 SAS 文件都存在于 SAS 逻辑库中。SAS 逻辑库用于组织、查找和管理 SAS 文件。在 SAS 中，通过该文件所在逻辑库及文件名来使用 SAS 文件。

SAS 数据集由 SAS 创建和管理，是 SAS 存储和处理数据的主要方式。根据其文件是否

包含数据值分为 SAS 数据文件和 SAS 视图。SAS 数据文件和 SAS 视图可以简单理解为与我们经常使用的数据库管理系统中的表和视图。关于 SAS 逻辑库和数据集将在第 2 章进行更详细的讲解。

SAS 目录 (Catalog) 是一种特殊的 SAS 文件, 以目录项的形式存储多种不同类型信息。一个 SAS 目录可包含多种类型的目录项, 这些目录项包含系统信息 (例如功能键定义) 和应用程序信息 (例如窗口定义、帮助窗口、格式 (Format)、读入格式 (Informat)、宏 (Macro) 或图形输出)。

下面通过“SAS 资源管理器”窗口浏览 SAS 逻辑库及 SAS 文件。启动 SAS 窗口环境, 在“SAS 资源管理器”窗口双击“逻辑库”→ sashelp。如图 1.5 所示的界面给出了当前环境下 SAS 系统中的逻辑库: Maps、Mapsgfk、Mapssas、Sashelp、Sasuser 和 Work。逻辑库 Sashelp 中的 SAS 文件如图 1.6 所示, 图标为数据集, 图标为 SAS 目录。SAS 目录还可打开显示更小的单元目录项。

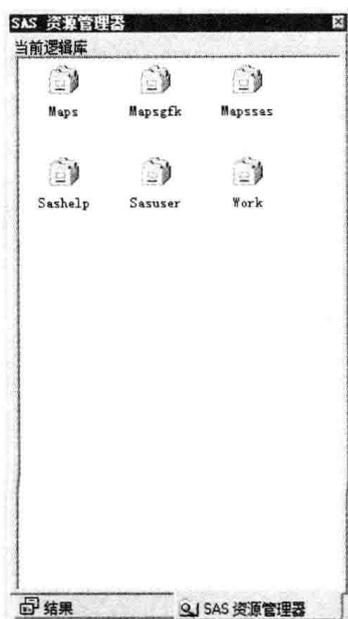


图 1.5 SAS 逻辑库

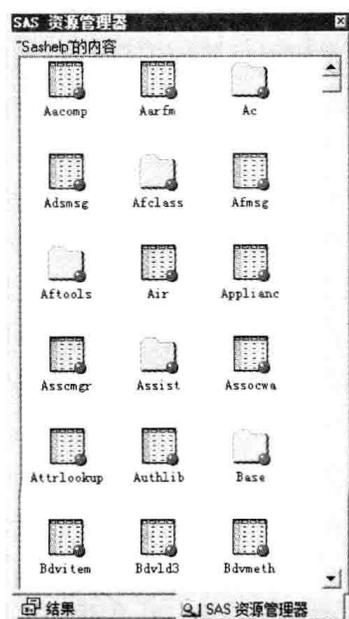


图 1.6 SAS 数据集和目录

SAS 除了可以处理 SAS 文件外, 还可以处理外部文件和数据库管理系统 (Database Management System, DBMS) 文件。SAS 处理的外部文件指由操作系统管理和维护的数据或文本文件。外部文件通常用于存储 SAS 需要处理的原始数据、SAS 程序语句和过程运行结果 (HTML, PDF 格式) 等, SAS 有时也会将一些结果写入外部文件。同时, SAS 还可以通过特定的 SAS/ACCESS 接口软件从其他厂商的软件系统 (例如数据库管理系统 (DBMS)) 文件中读取和写入数据。通过 SAS/ACCESS 接口软件建立到 DBMS 的 SAS 逻辑库后, SAS 软件可以像访问 SAS 数据集一样访问 DBMS 中的表。

1.5 一个简单的 SAS 程序

在了解了 SAS 窗口环境并简单了解了 SAS 逻辑库、数据集这些基础概念后，就可以在 SAS 窗口环境编辑一个简单的 SAS 程序，提交运行并查看结果了。SAS 程序由 DATA 步和过程步组成。DATA 步由关键字 DATA 开始，过程步由关键字 PROC 开始。当然 SAS 程序还可以包括宏语言，这个会在本篇后面的章节中专门介绍。

启动 SAS 窗口环境，在“程序编辑器”窗口输入下面的代码。

```
libname saslib base 'c:\sas\data';

data saslib.Inventory;
    input Product_ID $ Instock Price;
    datalines;
P001R 12 125.00
P003T 34 40.00
P301M 23 500.00
PC02M 12 100.00
;

proc print data= saslib.inventory;
run;
```

代码前面的部分之前已经介绍过。LIBNAME 语句定义物理路径在 c:\sas\data 的 SAS 逻辑库 saslib。DATA 步创建存储在逻辑库 saslib 下的 SAS 数据集 Inventory，其中包含 3 个变量（列），分别为 Product_ID、Instock 和 Price。在 DATA 步中 SAS 会读取紧接着 DATALINES 语句并以分号结束的 4 行输入数据，每行数据按顺序赋值给前面 3 个变量。PRINT 过程会在结果查看器中打印 DATA 步创建的数据集 Inventory 中的数值。

在输入代码后，保证该“程序编辑器”窗口为活动窗口，然后选择菜单“文件”→“保存”，单击“工具栏”的工具项或按快捷键 Ctrl+S 保存所输入的 SAS 程序语句为“测试程序”。当前的窗口环境如图 1.7 所示。这时要保证 c:\sas\data 文件夹存在，如果没有，在对应的目录创建一个，否则在“日志”窗口会提示错误及错误信息。选择菜单“运行”→“提交”，或单击“工具栏”中的工具项、按快捷键 F3 提交代码。

默认情况下代码提交执行后，包含运行结果的文件 HTML 会产生，并自动显示为当前活动窗口，而主窗口左侧则会显示结果列表，如图 1.8 所示。

在图 1.8 中，数据集 Inventory 的观测值和变量以表的形式展现在了 HTML 格式的“结果查看器”中。其中，“SAS 系统”字样是 SAS 系统默认标题。可以在 PRINT 过程之前使用 TITLE 语句指定输出标题或在 PRINT 过程中指定标题，通常会指定有意义的文字，比如“仓库库存”。指定的标题在整个 SAS 会话期间一直有效，如果要使用其他输出标题或恢复默认标题，可以再次使用 TITLE 语句指定。此外，还可以指定结果的多级标题和脚注，这里不赘述。

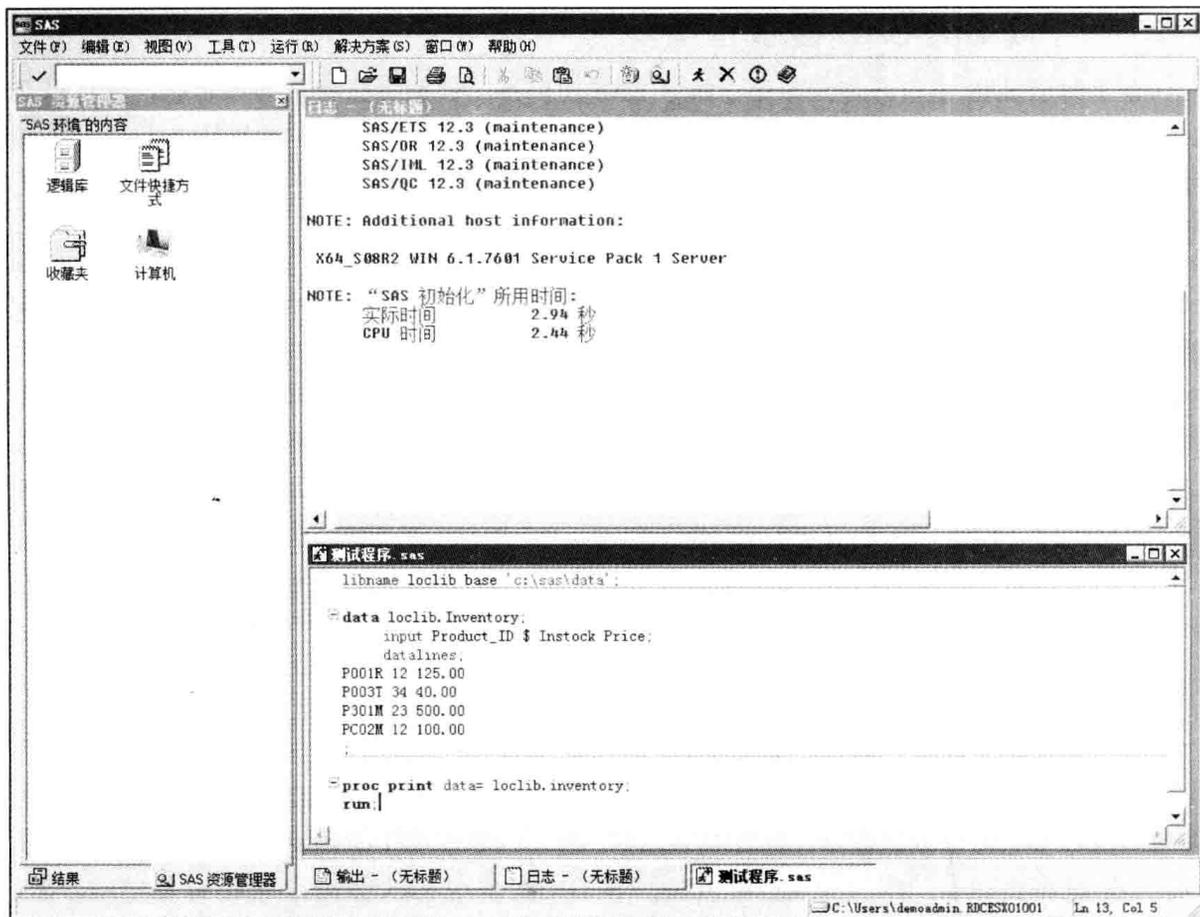


图 1.7 编辑 SAS 程序代码

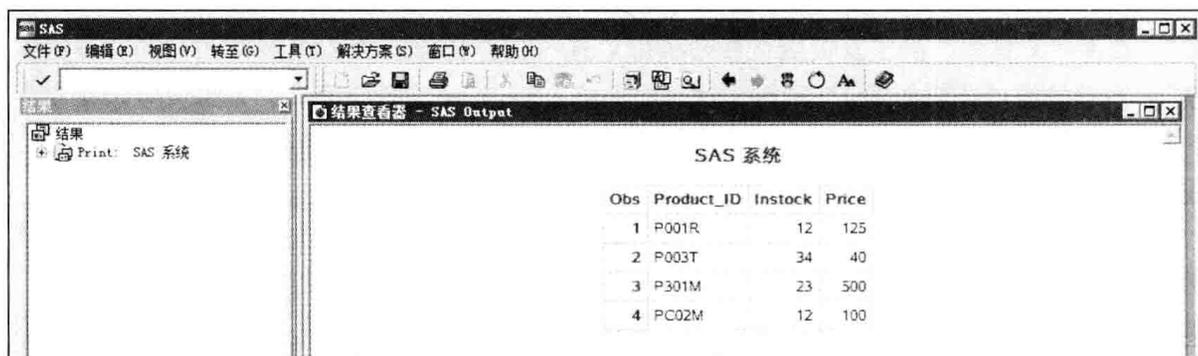
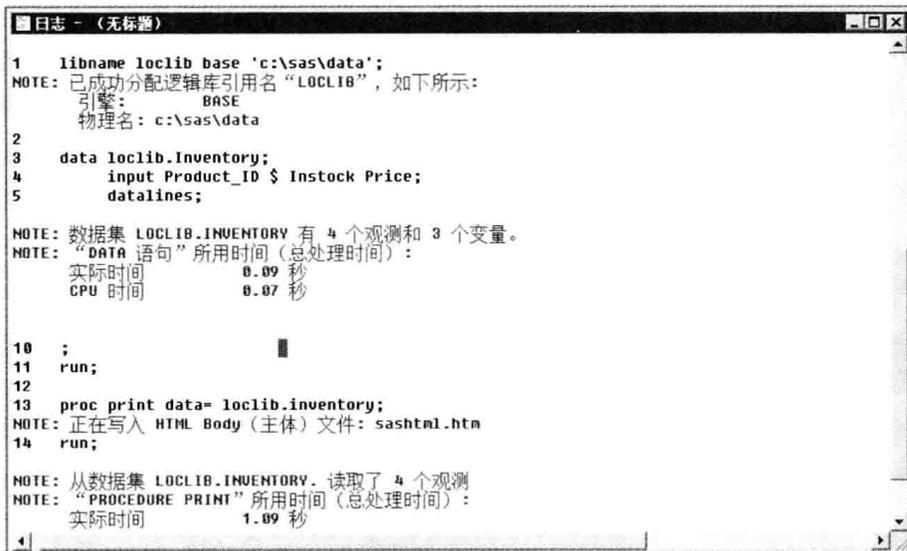


图 1.8 SAS 程序运行结果

可以通过“结果”窗口的条目在“结果查看器”的 HTML 文件中导航。在图 1.8 所示的“结果”窗口展开“Print: SAS 系统”，并单击 HTML 条目，该条目对应的结果会自动展示在“结果查看器”的当前位置。

在代码运行过程中，运行日志会显示在“日志”窗口。单击窗口条的“日志 - (无标题)”可显示日志信息，如图 1.9 所示。日志中给出了语句执行结果、DATA 步中生成数据集的观测数（行）和变量数（列），以及 PROC 步读取的观测数，还有运行所有 DATA 步和 PROC 步所花费的实际时间和 CPU 时间。“日志”窗口还可显示程序中的 PUT 语句输出，这个在本示例中没有涉及。



```

日志 - (无标题)
1  libname loclib base 'c:\sas\data';
NOTE: 已成功分配逻辑库引用名“LOCLIB”，如下所示：
      引擎：      BASE
      物理名： c:\sas\data
2
3  data loclib.inventory;
4  input Product_ID $ Instock Price;
5  datalines;

NOTE: 数据集 LOCLIB.INVENTORY 有 4 个观测和 3 个变量。
NOTE: “DATA 语句”所用时间（总处理时间）：
      实际时间      0.09 秒
      CPU 时间      0.07 秒

10 ;
11 run;
12
13 proc print data= loclib.inventory;
NOTE: 正在写入 HTML Body (主体) 文件: sashtml.htm
14 run;

NOTE: 从数据集 LOCLIB.INVENTORY. 读取了 4 个观测
NOTE: “PROCEDURE PRINT”所用时间（总处理时间）：
      实际时间      1.09 秒

```

图 1.9 SAS 代码运行日志信息

注意 要养成提交代码运行后首先查看“日志”窗口信息的良好习惯，检查日志中是否有错误或警告信息。很多时候，特别是当提交代码量较大时，即使前面的 DATA 步或 PROC 步运行失败，后续代码语句仍然会继续运行，但是可能会导致不正确的结果。所以必须检查日志中是否有需要注意的错误或警告信息。

接下来看看“输出”窗口。“输出”窗口展示 SAS 会话期间 SAS 语句的列表输出。SAS 窗口环境默认不产生列表输出。尝试在上述代码的基础上稍作修改：改变输出标题和产生列表输出，并在 PRINT 过程后关闭列表输出。修改后的代码如下：

```

libname saslib base 'c:\sas\data';

data saslib.inventory;
  input Product_ID $ Instock Price;
  datalines;
P001R 12 125.00
P003T 34 40.00
P301M 23 500.00
PC02M 12 100.00
;

```

```
ods listing;

title '仓库库存';
proc print data= saslib.inventory;
run;

ods listing close;
```

提交运行后，在默认打开的“结果查看器”中，HTML 文件在包含本次代码输出的同时，还包含了上次提交代码产生的输出。在“结果”和“结果查看器”窗口都可以看到第二次运行结果的标题为我们在程序代码里指定的“仓库库存”。展开结果窗口的叶子节点，还可以看到第二次提交的代码产生了两类输出：HTML 输出和列表输出。单击最后一个条目或单击窗口条的“输出 - (无标题)”打开所产生的列表输出，可以看到数据集数据也输出到了输出窗口，如图 1.10 右下窗口所示。

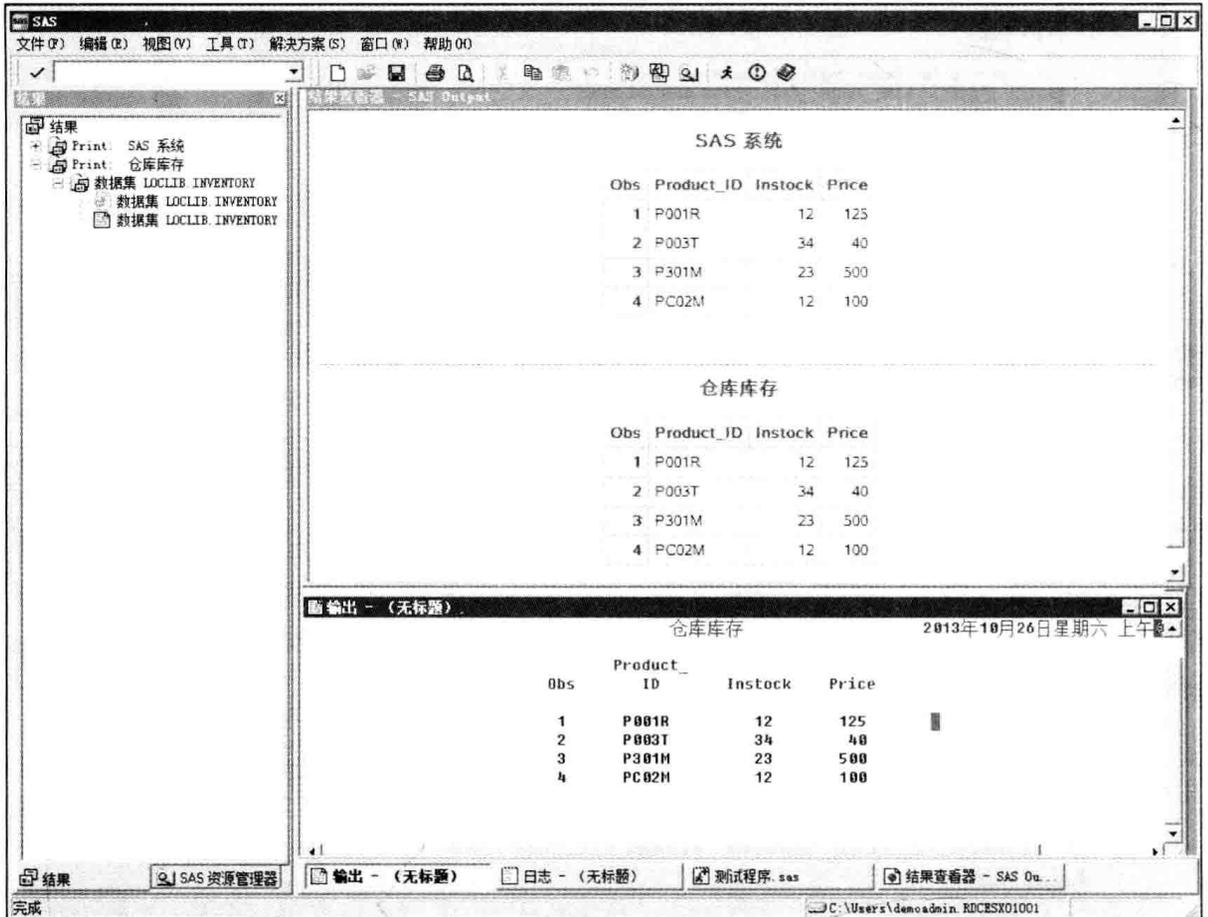


图 1.10 定制标题和列表输出

到这里，已经了解如何在 SAS 窗口环境中编辑一个简单的 SAS 程序、提交 SAS 程序并

检查 SAS 程序的运行日志、查看结果和输出。下面来看看基于 SAS 9.4 发布的另一可以交互方式提交 SAS 程序语句的 Web 应用 SAS Studio。

1.6 SAS Studio

SAS Studio 是基于 HTML5 客户端 / 服务器结构的 Web 应用。通过 SAS Studio, 用户能够以使用与 SAS 窗口环境类似的方式提交 SAS 程序语句, 运行并获取结果, 同时它还提供交互方式指导用户完成分析过程。

用户通过 SAS Studio 编写的代码或图形界面产生的分析过程会提交到本地或远程的 SAS 软件上执行, 结果返回 SAS Studio 客户端。这种特性会使 SAS Studio 承担 SAS 的 PaaS (Platform as a Service, 平台即服务) 中重要角色, SAS Studio 也会集成到 SAS 云基础设施中。

该产品有着丰富的操作界面, 这里简单了解一下。SAS Studio 3.1 的窗口如图 1.11 所示, 其中包含以下 3 个部分:

- ❑ 窗口顶部, 包含在 SAS Studio 中开发的应用程序名称和应用程序按钮。应用程序名称如图 1.11 中的“程序 1.sas”、“分布分析 1”和“直方图 1”。
- ❑ 窗口的左侧是具有多个可折叠条目的导航面板。导航面板提供的条目包含“搜索”、“文件夹”、“任务”、“代码段”、“逻辑库”和“文件快捷方式”。
- ❑ 右侧窗口(工作区)包括主要的选项卡, 可以显示 SAS 表, 文本文件(例如, SAS 程序文件)、任务等, 显示哪一个取决于当前执行的操作。



图 1.11 SAS Studio 开发 SAS 程序

通过这些窗口，用户可搜索文件夹、文件、SAS 逻辑库、SAS 表、表的列，还可以查看用户可用的文件夹和文件、代码段、SAS 逻辑库及逻辑库中的表、文件快捷方式等，并且可使用“任务”窗口提供的那些预先定义的任务，以图形界面操作交互方式实现多种分析过程。如图 1.12 所示是逻辑库 sashelp 中的 shoes 数据集的列 Sales 生成直方图的例子。选择“任务”→“图形”→“直方图”，在中间子窗口选择数据源和分析变量，最右侧子窗口会显示自动生成的代码。该代码可保存为代码段或直接提交到 SAS 服务器执行生成直方图。

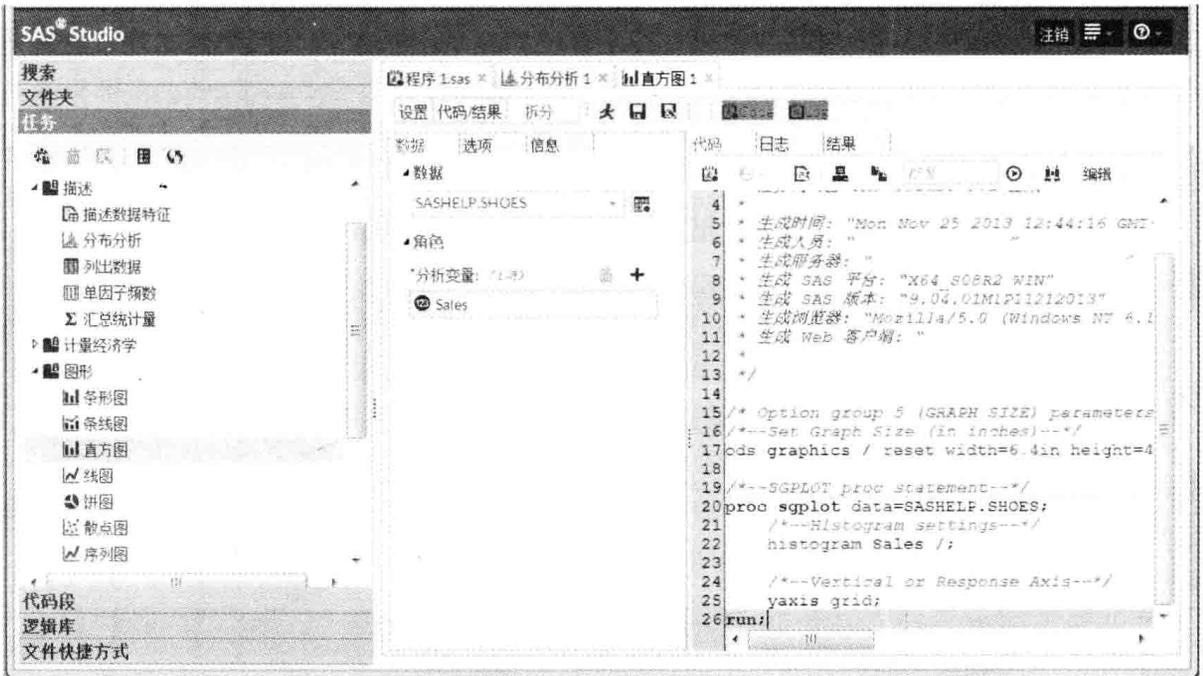


图 1.12 SAS Studio 生成直方图

1.7 本章小结

学习了本章内容后，相信读者对 SAS 软件、SAS 在 Windows 和 UNIX 环境下的各种启动方式，以及对 SAS 编程 IDE 环境（即 SAS 窗口环境和 SAS Studio）已经有了一些初步的了解，这会为后面的学习打下基础，可方便读者更加深入地学习 SAS 软件，并熟练使用 SAS 窗口环境完成数据处理和分析任务等。

读取外部数据到 SAS 数据集

SAS 提供丰富的数据处理和分析方法来解决各种商业问题，但在使用这些方法之前，往往需要首先将各种形式的数据转换成 SAS 数据集。如图 2.1 所示是在 SAS 中从原始数据到最后生成有价值信息的过程示意图。可以看出，在这个过程中，首要工作是将需要分析的原始数据转换成 SAS 数据集，然后才是运用各种 PROC 步对数据集里的数据进行处理和分析，最后将分析结果以适当的形式展现出来。

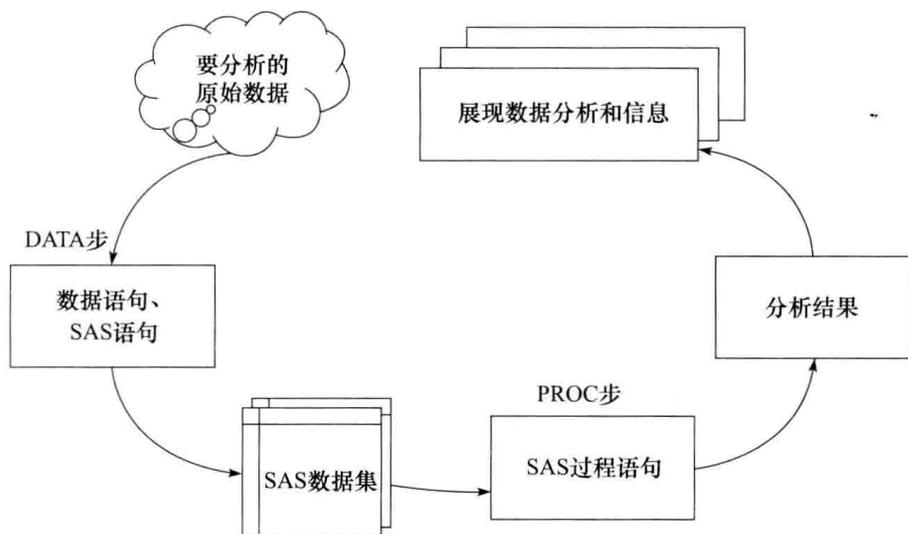


图 2.1 数据分析处理过程

本章在介绍 SAS 编程过程中经常使用的 SAS 逻辑库、SAS 数据集、系统选项以及 SAS

程序结构等基本概念之后，将会重点讲解如何开发 SAS 程序读取外部数据源中的原始数据，创建 SAS 数据集。最后简单介绍在 SAS 程序开发中常见的几种错误现象及其处理方法。

2.1 SAS 编程基本概念

在 SAS 系统中，SAS 程序是用来获取外部数据、处理和管理数据，并对其进行分析预测和优化，从而生成信息报告的重要工具。在学习开发 SAS 程序之前，首先需要理解两个基本概念：SAS 数据集和逻辑库，包括它们的命名、引用、类别等；然后得了解 SAS 中会经常使用的系统选项，这些系统选项让 SAS 的分析处理功能既灵活又强大。

2.1.1 SAS 逻辑库

SAS 逻辑库是一个或多个 SAS 文件的集合，用于组织、查找和管理 SAS 文件。SAS 逻辑库管理的 SAS 文件包括 SAS 数据集、SAS 目录、已编译的 SAS 程序，以及多维数据库文件等。在 Windows 和 UNIX 环境中，SAS 逻辑库通常是包含在同一个文件夹或目录下的一组 SAS 文件，其他文件也可以存储在该文件夹或目录下，但只有具有 SAS 文件扩展名的那些文件会被认为是该 SAS 逻辑库的一部分。在其他操作系统下，SAS 逻辑库有不同的实现方式，但通常都对应于当前操作系统用来访问和存储文件的组织级别。例如在 z/OS (OS/390) 操作系统下，SAS 逻辑库是只能存储 SAS 文件的特殊格式化了的主机数据集。尽管实现方式不一样，但在 SAS 支持的各种操作系统下，SAS 逻辑库的使用方式是相同的。

还有一种 SAS 逻辑库，叫作元数据边界逻辑库 (metadata-bound library)，是绑定在由元数据提供安全访问控制的对应表对象上的物理逻辑库。元数据边界逻辑库里的每个物理表都有指向特定元数据对象的信息，同时还创建了物理表和元数据对象之间的安全性绑定。该绑定保证了用户在访问该物理表时，SAS 强制执行元数据层权限要求，从而避免从操作系统直接访问该物理表导致的安全问题。这种 SAS 逻辑库在 SAS 智能分析平台中可用，在本章不作介绍。

1. 逻辑库关联

可以通过 LIBNAME 语句、LIBNAME 函数、使用“新建逻辑库”窗口或操作环境命令来定义逻辑库，并将 SAS 逻辑库与对应的逻辑库引用名关联起来。之后便可以通过该逻辑库引用名来读取、写入并更新 SAS 逻辑库中的 SAS 文件。使用 LIBNAME 语句定义 SAS 逻辑库的简化语法如下：

```
LIBNAME 逻辑库引用名 <逻辑库引擎> '逻辑库物理位置';
```

(1) 逻辑库引用名

在定义 SAS 逻辑库时需要指定逻辑库引用名，临时逻辑库 WORK 除外。SAS 逻辑库引用名的命名规范如下：

- 最大长度是 8 个字符。
- 必须以字母 (从 A ~ Z，大小写均可) 或下划线 (_) 开始。

□ 可以是数字 (0 ~ 9)、字母和下划线 (_) 的任意组合。

例如, 下面的 SAS 语句定义了 SAS 逻辑库, 其引用名为 saslib。

```
libname saslib base 'c:\sas\data';
```

 **注意** SAS 逻辑库与 SAS 逻辑库引用名是两个比较容易混淆的概念。SAS 逻辑库是 SAS 文件的集合, SAS 文件是其组成部分; 而 SAS 逻辑库引用名是我们定义逻辑库时赋予这个逻辑库的引用名, 并且在以后可以通过该逻辑库引用名来访问逻辑库中的 SAS 文件。永久逻辑库 (即其中的 SAS 文件) 会一直存在, 而通常 SAS 逻辑库引用名仅在当前 SAS 会话中有效, 除非使用 “新建逻辑库” 窗口指定逻辑库时勾选了 “启动时启用” 选项。永久逻辑库和 “新建逻辑库” 窗口, 在后面都有介绍。

(2) 逻辑库引擎

SAS 逻辑库引擎是 SAS 软件和 SAS 逻辑库之间的接口软件组件, 每个 SAS 逻辑库都关联一种逻辑库引擎。逻辑库引擎识别逻辑库中的文件并以 SAS 可理解的格式将文件内容呈现给 SAS。SAS 提供多种引擎以管理多个格式的数据。通过这些引擎 SAS 可以进行如下操作: 存储和访问磁盘文件, 将数据从物理位置放入内存中, 读取其他软件产生的数据库文件, 以及在不同操作系统之间迁移 SAS 文件等。

逻辑库引擎可分为原生逻辑库引擎和接口逻辑库引擎。原生逻辑库引擎也就是默认的 Base 引擎, 访问由 SAS 创建和处理的 SAS 文件。在创建新逻辑库时如果不指定引擎, SAS 会自动选择 Base SAS 引擎。该引擎根据自身版本的不同, 可处理 SAS 7、SAS 8 和 SAS 9 文件, 这些文件对应的版本引擎名称分别为 V7、V8 和 V9。大多数情况下, SAS 9 可以直接处理 SAS 8、SAS 7 和 SAS 6 创建的 SAS 文件, 不需要对其进行转换。关于版本兼容性可能存在的问题及对应解决方法, 可参考 SAS 帮助文档学习。

接口逻辑库引擎用来访问由其他软件系统 (例如关系型数据库系统、ERP 系统等) 管理的数据。接口逻辑库对用户不透明, 需要显式指定引擎名称, 并且需要相应的 SAS/ACCESS 软件许可, 而且通常还需要安装相应的数据库管理系统的客户端软件。例如要访问 Teradata 数据库管理系统中的数据文件, 要有 SAS/ACCESS to Teradata Interface 的许可, 并且在定义逻辑库时, LIBNAME 语句中要指定对应的逻辑库引擎为 Teradata 数据库对应的引擎, 同时还要安装了 Teradata 提供的客户端软件。

下面给出了定义原生逻辑库和接口逻辑库示例。

□ Base 引擎相关代码如下 (语句中 base 选项可省略):

```
libname saslib base 'c:\sas\data';
```

□ SAS/ACCESS to Teradata 引擎相关代码如下:

```
libname tdlb teradata server=tera2650 user=user1 password=password1 database=hps;
```

(3) 逻辑库物理位置

SAS 逻辑库物理位置是一个或多个操作系统能够识别的物理位置，或者是一个或多个已经定义了的其他 SAS 逻辑库。在上面给出的 Base 引擎示例中，逻辑库 saslib 的物理路径为 c:\sas\data。对于连接到数据库管理系统的 SAS 逻辑库，通常是通过一系列数据库连接选项指定要访问的数据库管理系统的信息。如在 SAS/ACCESS to Teradata 引擎示例中，通过数据库连接选项分别指定了 Teradata 数据库服务器的名称、使用的用户名、密码及数据库名称。

SAS 逻辑库还可以有多个物理位置。下面的代码示例给出了定义多个物理位置的 SAS 逻辑库 Y2014。这 3 段代码都能实现将逻辑库引用名 Y2014 与 4 个物理位置 c:\sas\data\quater1、c:\sas\data\quater2、c:\sas\data\quater3 和 c:\sas\data\quater4 相关联。

□ 代码 1:

```
libname Y2014 ('c:\sas\data\quater1' 'c:\sas\data\quater2'
'c:\sas\data\quater3' 'c:\sas\data\quater4');
```

□ 代码 2:

```
libname Q1_2004 'c:\sas\data\quater1';
libname Q2_2004 'c:\sas\data\quater2';
libname Q3_2004 'c:\sas\data\quater3';
libname Q4_2004 'c:\sas\data\quater4';
libname Y2014 (Q1_2004 Q2_2004 Q3_2004 Q4_2004);
```

代码 3:

```
libname Q2_2004 'c:\sas\data\quater2';
libname Q3_2004 'c:\sas\data\quater3';
libname Q4_2004 'c:\sas\data\quater4';
libname Y2014 ('c:\sas\data\quater1' Q2_2004 Q3_2004 Q4_2004);
```

当一个物理位置下的空间不够时，定义多个物理位置的 SAS 逻辑库非常有用。这样，在写程序时只需要使用一个 SAS 逻辑库引用名，当前面的物理路径空间用尽时，SAS 会自动将写入的 SAS 文件存储到其他物理路径。

2. 永久和临时 SAS 逻辑库

SAS 逻辑库通常为永久数据库。永久 SAS 逻辑库存储在计算机的固定存储介质上，当 SAS 会话终止时不会被删除，其中的 SAS 文件可以在后续 SAS 会话中继续使用。当使用永久 SAS 逻辑库中的文件时，通常需要指定逻辑库引用名作为两层 SAS 文件名的第一部分，并且要告诉 SAS 该文件的存储位置，例如 saslib.Inventory，表明读取或写入 SAS 逻辑库 saslib 中的 Inventory 文件、数据集或目录 (Catalog) 等，至于 Inventory 具体指的是哪种 SAS 文件，与所使用的上下文环境有关。

同时 SAS 还提供了一种在 SAS 会话或作业运行过程中存储临时数据和文件的临时逻辑库，其引用名为 WORK。逻辑库 WORK 不需要显式指定，且仅在当前 SAS 会话或作业执行过程中存在。逻辑库 WORK 中的文件在该 SAS 会话期间可用于任何 DATA 步或 SAS 过程，但如果

SAS 会话正常结束，WORK 库中的文件在 SAS 会话结束时会被自动删除。一般情况下，可以通过指定一级名称来读写这个逻辑库中的 SAS 文件，同样也可以使用二级名称。例如，要引用临时逻辑库中的 SAS 文件 Inventory，直接使用 Inventory 和使用 work.Inventory 的效果一样。

在开发 SAS 程序时，如果一次分析包含多个 PROC 步，通常会将前一个 PROC 步产生的中间数据或文件放入临时逻辑库中，供后面的分析过程使用。在分析完成时，这些中间数据或文件会自动清除。当然，良好的开发风格应该是在完成任务后，通过代码显式地删除所产生的临时数据。

3. SAS 系统逻辑库

SAS 提供了 4 个特殊的系统逻辑库：WORK、user、sashelp 和 sasuser。WORK 是临时逻辑库，前面已经介绍过，其他 3 个都是永久逻辑库。

user 逻辑库可以使用 LIBNAME 语句、LIBNAME 函数、系统选项 USER= 或操作系统显式指定。指定 user 逻辑库后，可以使用一级名称读取该逻辑库中的文件，就像引用临时逻辑库中的文件一样。一旦定义了 SAS 逻辑库 user，在 SAS 程序中使用一级名称读取或写入任何 SAS 文件时，SAS 都会在该 user 逻辑库对应的物理位置查找或写入相应的 SAS 文件。这时，如果要引用 WORK 逻辑库中的文件，必须指定带有 WORK 逻辑库引用名的二级名称。当 SAS 会话结束时，存储在逻辑库 user 里的文件不会被删除。

sashelp 逻辑库包含一组用于控制 SAS 会话各方面信息的目录（Catalog）和其他文件。该逻辑库中存储的这些目录和文件适用于任何使用该 SAS 系统的用户。用户的个性化设置会存储在 sasuser 逻辑库中。如果除了安装 Base SAS 软件外，还装了 SAS 的其他产品，那么这些产品需要用到的一些目录也会包含在 sashelp 逻辑库中。

sasuser 逻辑库包含能够定制 SAS 特征以满足特定要求的 SAS 目录。如果默认的 sashelp 逻辑库对一些应用程序不适用，那么可以修改它们并将这些个性化的设置保存在 sasuser 逻辑库中。例如，在 SAS 里，可以在名称为 sasuser.profile 的个人 Profile 目录中存储个人默认的功能键设置或窗口属性。

2.1.2 SAS 数据集

SAS 数据集是存储在 SAS 逻辑库中、由 SAS 创建和处理的 SAS 文件，是 SAS 存储数据的主要方式。SAS 数据集包含以表的观测（行）和变量（列）为形式存在的数据值，以及用以描述变量类型、长度和创建该数据集时所使用的引擎等信息的描述信息。根据其是否包含真正的数据值，SAS 数据集可分为 SAS 数据文件和 SAS 视图。SAS 数据文件包含数据和描述信息，在逻辑库中的成员类型是 DATA；而 SAS 视图不包含数据值，是指向其他数据源的虚数据集，成员类型是 VIEW。下面分别介绍 SAS 数据集的文件内容、命名，各种 SAS 数据文件和 SAS 视图，以及它们的创建方式。

1. 数据集文件

如图 2.2 所示给出了 SAS 数据集的逻辑组件，这些组件可能分布在操作系统下的不同文件中。



图 2.2 SAS 数据集逻辑组件

下面来具体看看图 2.2 中的各个组件。

- ❑ **描述信息:** 描述了 SAS 数据集自身及其变量的属性，包括观测数、观测长度、该数据集上次的修改日期等其他信息。其变量的描述信息包括名称、类型、长度、输入输出格式、标签，以及是否已经为该变量建立索引等属性。
- ❑ **数据值:** 以矩形表的形式排列。一个数据集可包含若干个观测（也称为行），每个观测通常由一个或多个变量（也称为列）值组成。
- ❑ **索引:** 是单独的 SAS 文件，可以为 SAS 数据文件创建索引，以提供对指定观测的直接访问。索引文件与其数据文件有着相同的名称，但成员类型为 INDEX。索引可提供对指定观测更快的访问，尤其是对较大的数据集而言。
- ❑ **扩展属性:** 是定义在数据集或变量之上的元数据。扩展属性使用 DATASETS 过程创建，表示为 <名称 - 值> 对。

下面以一个简单的 SAS 数据集为例，来理解数据集的描述信息、观测和变量。

在 SAS 窗口提交如下代码生成数据集：

```
libname saslib 'c:\sas\data';

data saslib.Inventory;
    input Product_ID $ Instock Price;
    datalines;
P001R 12 125.00
P003T 34 40.00
P301M 23 500.00
PC02M 12 100.00
;
run;
```

使用 CONTENTS 过程打印数据集的属性信息，代码如下：

```
proc contents data=saslib.inventory;
```

```
run;
```

CONTENTS 过程生成的结果如图 2.3 所示。其中包含了上面提到的数据集信息、主机相关信息和变量信息等。

CONTENTS PROCEDURE				引擎/主机相关的信息		按字母排序的变量和属性列表			
数据集名	SASLIB INVENTORY	观测	4	数据集页面大小	65536	#	变量	类型	长度
成员类型	DATA	变量	3	数据集页数	1	2	Instock	数值	8
引擎	V9	索引	0	首数据页	1	3	Price	数值	8
创建时间	2013-11-13 02:15:55	观测长度	24	每页最大观测数	2715	1	Product_ID	字符	8
上次修改时间	2013-11-13 02:15:55	删除的观测	0	首数据页的观测数	4				
保护		已压缩	NO	数据集修复数	0				
数据集类型		已排序	NO	ExtendObsCounter	YES				
标签				文件名	c:\sas\data\inventory sas7bdat				
数据表示法	WINDOWS_64			创建版本	9 0401M0				
编码	euc-cn Simplified Chinese (EUC)			创建主机	X64_S08R2				

图 2.3 数据集描述信息输出

提交 PRINT 过程代码，打印数据集信息。

```
proc print data=saslib.inventory noobs;
run;
```

上面代码打印的数据集如图 2.4 所示。

该数据集包含 4 个观测，每个观测表示一种产品的各类信息。这里有 3 个变量：Product_ID、Instock 和 Price，分别表示产品编号、库存数和价格。其中 P001R、12、125 等均为数据值。

Product_ID	Instock	Price
P001R	12	125
P003T	34	40
P301M	23	500
PC02M	12	100

图 2.4 数据集数据值

2. 数据集命名

每个 SAS 数据集的完整名称如下：libref.SAS-data-set.membertype。共 3 个组成部分，从左到右依次为逻辑库引用名、数据集名称和成员类型。在引用数据集时，通常会指定前两个，SAS 会根据上下文环境，例如该数据集出现的位置或数据集的自描述信息，来确定第三个。逻辑库引用名是与 SAS 数据集所在物理位置相关联的 SAS 逻辑库名。当创建新数据集时，逻辑库引用名表明要将该数据集保存在哪里（位置）。当引用 SAS 数据集时，逻辑库引用名会告诉 SAS 在哪个逻辑库中找到该数据集。

数据集名称遵守的 SAS 命名规则如下：

- 最大长度为 32 字符。
- 必须以字母（从 A ~ Z，大小写均可）或下划线（_）开始。
- 可以是数字、字母和下划线（_）的任意组合。

成员类型由 SAS 指定，例如 SAS 数据文件的成员类型是 DATA，SAS 视图的成员类型是 VIEW。这些对开发 SAS 程序是透明的。

在程序语句中创建和使用 SAS 数据集时，根据数据集所在的逻辑库或要存储的逻辑库来确定使用一级或二级名称。一级名称只包含数据集名称，用于读写临时逻辑库 WORK 中

的数据集，或当逻辑库 user 被指定时读写逻辑库 user 中的数据集。二级名称由逻辑库引用名和数据集名称组成，形式为 libref.SAS-data-set，访问除逻辑库 user 之外的其他永久逻辑库中的数据集时，均需要使用二级名称。



注意 前面介绍过 SAS 数据文件和视图都是 SAS 数据集。SAS 不允许在相同的逻辑库中存在数据集名称相同的 SAS 数据文件和视图。因为从语法上来讲，同一程序语句可以同时接受 SAS 数据文件和 SAS 视图，SAS 不能从程序语句判断需要处理的是哪一个文件。

虽然在访问 WORK 或 user 逻辑库中的数据集时可以用一级名称，即可省略逻辑库引用名，但为了保持良好 SAS 代码编写风格，不建议省略。

3. 变量属性

SAS 数据集变量的属性包括变量名、类型、长度、输出格式 (format)、输入格式 (informat) 和标签 (label)。输出格式、输入格式和标签是变量的可选属性。

每个变量的变量名必须遵守的 SAS 命名规范如下：

- ❑ 最大长度为 32 字节。
- ❑ 必须以字母 (从 A ~ Z，大小写均可) 或下划线 (_) 开始。
- ❑ 可以是数字、字母和下划线 (_) 的任意组合。

变量的类型是字符型或数字型。字符型变量可包含任何值，而数字型变量只能包含数字值 (数字 0 ~ 9、=、-、点 (.) 和科学计数法的 E)。变量类型确定了变量的缺失值如何显示。字符型变量缺失值是空格，而数字型的变量缺失值是点 (.)。

SAS 以数字值存储日期和时间。默认情况下，SAS 的日期值指从 1960 年 1 月 1 日开始的天数，SAS 使用从凌晨开始的秒数存储时间值，SAS 的日期时间值 (datetime) 指从 1960 年 1 月 1 日开始的秒数。该开始日期也可以通过系统变量 YEARCUTOFF 指定为其他值。

下面提交如下代码来生成数据集 sales，并使用 CONTENTS 过程和 PRINT 过程分别打印该数据集的描述信息和数据值。

```
libname saslib 'c:\sas\data';

data saslib.sales;
  infile datalines dsd missover;

  input Emp_ID $ Dept $ Sales Date;
  format Sales COMMA10. Date yymmdd10.;
  informat Date date9.;
  label Emp_ID=" 员工 ID" Dept=" 部门 " Sales=" 销售数据 ";
  label Date=" 销售时间 ";
  datalines;
ET001,TSG,$10000,01JAN2012
```

```

ED002,, $12000,01FEB2012
ET004,TSG,$5000,02MAR2012
EC002,CSG,$23000,01APR2012
ED004,QSG,,01AUG2012
;
run;

proc contents data=saslib.sales;
run;

proc print data=saslib.sales noobs label;
run;

```

如图 2.5 所示为 CONTENTS 过程打印的部分结果，表示该数据集的变量属性，其中，Date 和 Sales 为数值型变量，Dept 和 Emp_ID 为字符型变量。如图 2.6 所示为 PRINT 过程的打印结果，可以看出，Dept（部门）的缺失值为空格，Sales（销售数据）的缺失值为点（.）。

按字母排序的变量和属性列表						
#	变量	类型	长度	输出格式	输入格式	标签
4	Date	数值	8	YYMMDD10.	DATE9	销售时间
2	Dept	字符	8			部门
1	Emp_ID	字符	8			员工ID
3	Sales	数值	8	COMMA10.	DOLLAR10	销售数据

图 2.5 变量属性

员工ID	部门	销售数据	销售时间
ET001	TSG	10,000	2012-01-01
ED002		12,000	2012-02-01
ET004	TSG	5,000	2012-03-02
EC002	CSG	23,000	2012-04-01
ED004	QSG	.	2012-08-01

图 2.6 打印数据集

变量的长度与类型有关。字符变量的长度可以在定义时给出，否则其长度为第一次赋值时值的长度，最大长度可到 32K。数字型变量的默认长度是 8 个字节，也可以指定不同的长度。

除了名称、类型和长度外，还可以定义变量的输出格式、输入格式和标签，这些都是可选属性。

格式（format）会影响数据值输出的方式。SAS 提供了各种字符、数字和日期时间格式。例如，为了将 23 000 显示为 23 000，必须使用 COMMAw.d 形式的输出格式。其中 w 表示最大宽度，d 为小数位数。比如，在图 2.5 中，Date 变量的输出格式为“YYMMDD10.”，打印时该变量的形式则为 YYYY-MM-DD（例如 2012-01-01）。Sales 的输出格式为“COMMA10.”，对应的数据输出形式则为 10 000。还可以创建并存储自定义的格式，具体在第 5 章介绍。

输入格式（informat）指定数据值以特定的格式读入，从而成为标准的 SAS 值。在读取包含字母或其他特殊字符的数字值时必须使用输入格式。例如，需要把输入值“\$23000”读取为数字型的变量，则必须使用输入格式 DOLLARw.d 才能正确读入。自定义的格式也可以用作输入格式。

SAS 提供了丰富的输入输出格式用于从外部文件读取各种日期格式和显示各种日期格式，以满足对各种日期格式的需要。在上面的示例中使用输入格式“DATE9.”读入了“01JAN2012”形式的日期，输出时使用的是输出格式“YYMMDD10.”，从而将存储的数字显示为“2012-01-01”。

变量都可以有标签 (label)。标签通常是描述该变量的文本, 最大长度为 256 个字符。默认情况下, 报表以变量名来标识变量, 但是可以将一个标签分配给相应的变量来显示该变量的描述信息。上例中 Emp_ID 的标签为“员工编号”, Dept 的标签为“部门”, Date 的标签为“销售时间”, Sales 的标签为“销售数据”。在代码中可以看到 PRINT 过程使用了 LABEL 选项, 这样一来, 打印的数据集表头将会使用各变量的标签而不是变量名称。

4. SAS 数据文件

与 SAS 视图相比较, SAS 数据文件是一种在其文件中包含数据的数据集。SAS 数据文件可以由 DATA 步创建, 其名称在 DATA 语句中指定, 还可以由 PROC 步创建, 其名称通常是在该 PROC 步语句或 PROC 步的 OUTPUT 语句中指定的。有时, 如果程序没有给输出数据集指定名称, SAS 会使用默认名称。

有两种类型的数据文件: 原生 SAS 数据文件和接口 SAS 数据文件。原生数据文件是 SAS 格式的文件, 用来存储 SAS 格式的数据值和描述信息。接口数据文件是指数据以其他格式存在, 并且 SAS 可以通过 SAS/ACCESS 接口引擎访问的数据文件, 例如存在于 Oracle、DB2、Sybase、ERP 系统中的数据文件。SAS 通过 SAS/ACCESS 接口引擎来访问这些文件中的数据, 并将这些文件当作 SAS 数据集处理。

SAS 程序语句创建的是原生 SAS 数据文件还是接口 SAS 数据文件, 取决于该数据文件所属的逻辑数据库。如果该逻辑库是通过 Base 引擎定义的, 则所生成的数据文件是原生数据文件, 如果是通过 SAS/ACCESS 接口逻辑库定义的, 则所创建的是接口 SAS 数据文件。

使用 DATA 步创建 SAS 数据文件的语法如下:

```
DATA 数据集名称;
  ... SAS 语句...;
RUN;
```

其中, SAS 语句用于指定数据源。不同的数据源, SAS 语句也不尽相同。例如 DATALINES 语句表示从程序语句中读取数据, SET 语句读取指定的输入数据集, INFILE 语句读取指定的外部数据文件等。

上例中的 DATA 步给出了使用 DATALINES 语句读取列举输入数据的一个示例。该代码首先定义了一个物理路径为 c:\sas\data 文件夹的 SAS 逻辑库 saslib, 接着以 DATA 关键字开始的 DATA 步创建了存储在逻辑库 saslib 中的数据集 Inventory。

有些 PROC 步会在 PROC 语句或 PROC 步的 OUTPUT 语句中指定要输出的数据集。下面的代码使用 SUMMARY 过程对数据集 trucks 进行汇总, 该过程的 OUTPUT 指定将输出写入数据集 saslib.sumout 中。

```
libname saslib 'c:\sas\data';

proc summary data=saslib.trucks;
  var deaths;
  output out=saslib.sumout n=n;
run;
```

(1) SAS 数据文件观测数

观测数是 SAS 数据文件的一个重要属性。SAS 数据文件的观测数是文件中当前观测（行）和已删除观测的总和。可以通过执行 CONTENTS 过程或 DATASET 过程的 CONTENTS 语句列出数据集的观测数，所列出的观测数是观测和已删除观测的总和。参考图 2.3，该数据集观测数在 CONTENTS 过程打印的第一个表格的第一列（“观测”项）中给出。

了解观测数有助于管理文件大小并评估磁盘空间要求。SAS 数据文件可计算的最大观测数由操作系统的长整型大小决定。

SAS 使用所在操作系统内部的长整型数来记录数据集的观测数。在 32 位操作系统中，长整型为 32 位，数据文件的最大观测数是 $2^{31}-1$ 。在 64 位操作系统中，当长整型长度为 64 位时，最大观测数是 $2^{63}-1$ 。在 64 位的操作系统中，长整型长度为 64 位时，SAS 数据文件不可能达到最大观测数，但是在 32 位操作系统中，达到最大值时有发生，一定要注意这点。同样需要注意的是，即使在微软的 Windows 64 位版本的操作系统中，其长整型数据类型实际使用的是 32 位模型，以便维持与 32 位应用的兼容性。从 SAS 9.3 开始，也可以通过设置选项 EXTENDOBSCOUNTER=YES 来扩展新输出 SAS 数据文件中的最大观测数。这样，即使在使用 32 位长整型存储观测数的操作系统中，所创建的 SAS 数据文件的最大观测数也能达到 $2^{63}-1$ 。

当达到最大观测数时，SAS 如何处理取决于该数据文件是否有索引，或是否使用了索引的完整性限制。

- ❑ 如果该 SAS 数据文件有索引或使用了索引的完整性限制（唯一键、主键和外键），SAS 会产生错误消息。可以删除索引或完整性限制并继续处理。但是因为文件超过了最大观测数，有些功能会受限制（受限制的功能会在下面介绍）。而且，如果要维持索引或完整性限制，必须重建该 SAS 数据文件。从 SAS 9.4 开始，当创建 SAS 数据文件时，默认会创建扩展的观测数。有兴趣的读者可查看 SAS 帮助文档获取更多信息。
- ❑ 如果没有使用索引，SAS 会继续后续处理并且接受额外的观测，不会有消息表明该 SAS 数据文件已经达到或超过最大观测数。

当文件超过最大观测数时，任何需要观测数的操作都不可用。例如，返回观测数的 SAS 过程（例如 PRINT 过程或 CONTENT 过程）会返回缺失值（.）；依赖于观测数的 SAS 过程（例如 SORT 过程或 COMPARE 过程）会返回不可预知的结果；当请求压缩文件时，压缩比例不可计算；不能创建索引或完整性限制等。为了重获这些功能，可以重新创建带扩展观测数的 SAS 数据文件。

(2) 审计追踪 SAS 文件

可以通过 DATASETS 过程对 SAS 数据文件创建审计追踪 SAS 文件，用来记录 SAS 数据文件的修改历史。每当观测被删除或更新时，谁在什么时候修改了什么的信息都会被写入审计文件。许多业务和组织为了安全都会要求审计追踪。审计追踪维护了历史信息，历史信息可用于追踪单块数据从录入数据文件开始到离开的所有信息。

在使用 APPEND 过程对数据文件进行附加操作时，如果因为完整性限制拒绝导致附加

操作失败，审计追踪将是 SAS 中存储这些失败观测的唯一技术。可以使用 DATA 步从审计跟踪文件中抽取失败或拒绝的观测，然后根据失败原因描述信息来纠正它们，最后将它们重新应用于该数据文件。

5. SAS 视图

SAS 视图本身并不存储数据值，它仅包含描述信息和从其他 SAS 数据集或从存储为其他软件厂商文件格式的文件中获取数据所需要的信息。SAS 视图的成员类型是 VIEW。

可使用 SQL 过程、ACCESS 过程或者 DATA 语句的 VIEW 选项来创建 SAS 视图。根据创建的方式，视图又分为 SQL 视图、接口 SAS 视图和 DATA 步视图。其中 SQL 视图和 DATA 步视图都称为原生视图，SAS/ACCESS 视图称为接口视图。关于 SQL 视图的内容在本书第 6 章介绍。

创建 DATA 步视图的语法如下：

```
DATA 数据集名称 / view= 数据集名称 ;
  ... SAS 语句...;
RUN;
```

与使用 DATA 步创建 SAS 数据文件一样，SAS 语句根据数据来源的不同会有所不同。在 DATA 步视图中可用的数据源可以是原始数据文件、SAS 数据文件、PROC SQL 视图、SAS/ACCESS 视图或其他数据库管理系统中的数据文件。下面给出了一段创建 DATA 视图的示例代码，数据源为 SAS 数据文件。

```
data saslib.invt_vw / view=saslib.invt_vw;
  set saslib.inventory;
run;
```

上面代码创建了 DATA 步视图 saslib.invt_vw，其数据来自于逻辑库 saslib 下的 Inventory 数据文件。

接口视图通过 SAS/ACCESS 创建，可读取第三方数据库管理系统（DBMS）的数据，例如 DB2 或 Oracle。其实，SAS/ACCESS 为这些第三方产品提供了 LIBNAME 引擎接口，对这些产品，建议使用 LIBNAME 和 SAS/ACCESS 对应的引擎来指定 SAS 逻辑库到 DBMS 数据，这比使用 ACCESS 过程创建接口视图更容易，也更有效。

使用 SAS 视图有如下优点：

- ❑ 可以节省磁盘空间，因为 SAS 视图不存储实际数据，仅仅存储去哪儿找到数据及数据如何格式化的指令。
- ❑ 因为数据总是在执行时才从 SAS 视图中获取，这样能保证输入的数据集总是当前的。
- ❑ SAS 视图可减少由于数据设计的改变对用户造成的影响。例如，可以改变存储在 SAS 视图里的查询信息而不必改变视图结果的特征。
- ❑ 使用 SAS/CONNECT 软件，SAS 视图可连接不同主机计算机上的 SAS 数据集，然后展示公司分布式数据的集成视图。

SAS 视图可用于以下操作：输入其他 DATA 步或 PROC 步，将数据迁移到 SAS 数据文

件或 SAS 支持的数据库管理系统中，使用 PROC SQL 与其他数据组合。

在选择使用 SAS 数据文件还是 SAS 视图时需要考虑以下方面：

- 数据文件会使用额外的磁盘空间，而 SAS 视图会占用额外的处理时间。
- 数据文件变量可在使用前排序和创建索引，而 SAS 视图在执行过程中只能以其存在的形式处理数据。

6. 特殊的数据集

还有一种特殊的数据集：`_NULL_`。如果想执行一个 DATA 步又不想创建 SAS 数据集，可以指定关键字 `_NULL_` 作为数据集名称。代码如下：

```
data _null_;
```

SAS 会执行该 DATA 步里面的语句但不会创建新数据集，不会有观测或变量写入任何数据集。如果一个 DATA 步的输出不需要存储为数据集，比如绘制报表，这种处理可更有效地利用计算机资源。

2.1.3 SAS 逻辑库和数据集管理

SAS 逻辑库和数据集可以通过 SAS 程序语句进行管理，例如 LIBNAME 语句、DATA 步、DATASETS 过程、APPEND 过程和 CONTENTS 过程等。DATA 步提供了许多功能对 SAS 数据集进行处理（本书后面的章节会具体介绍）。

1. LIBNAME 语句

LIBNAME 的 LIST 选项可以将一个或多个 SAS 逻辑库属性打印在日志中。其基本形式如下：

```
LIBNAME 逻辑库引用名 LIST;
LIBNAME _ALL_ LIST;
```

LIBNAME 的 CLEAR 选项用于清除一个或多个逻辑库引用名与 SAS 逻辑库之间的关联关系。清除了关联关系的逻辑库引用名在 SAS 会话中不再有效。其基本形式如下：

```
LIBNAME 逻辑库引用名 CLEAR;
LIBNAME _ALL_ CLEAR;
```

LIBNAME 还有更多对 SAS 逻辑库的管理功能，可参考 SAS 帮助文档进行学习。

2. CONTENTS 过程

CONTENTS 过程用于显示数据集的描述信息内容，并打印 SAS 逻辑库的目录。通常，CONTENTS 过程和 DATASETS 过程的 CONTENTS 语句相同。该过程的基本形式为：

```
PROC CONTENTS DATA=数据集名称;
RUN;
```

在前面的各节中已经使用 CONTENTS 过程显示了数据集属性内容，如图 2.3 所示。

3. DATASETS 过程

DATASETS 过程提供了强大的 SAS 数据文件管理功能，对数据集而言，除了可显示数

据集的描述信息外，还可用于追加观测、修改变量名、删除数据集等。对于该过程，可通过 SAS 帮助文档进行学习。

当需要修改一个数据集的变量属性时，使用 DATASETS 过程可以在不读取数据集观测的情况下直接修改变量属性。虽然也可以通过使用 DATA 步的 SET 语句基于一个数据集创建新的数据集，并为新数据集指定不同的属性来实现相同的功能，但这样 SAS 会读写所有的观测，造成资源浪费，当数据集较大时会非常耗时。

4. SAS 资源管理器

此外，我们也经常会通过 SAS 窗口环境的“SAS 资源管理器”来管理逻辑库以及逻辑库中的 SAS 文件。这里主要介绍 SAS 提供的对 SAS 逻辑库和数据集的一些实用管理操作，具体操作可以根据各个菜单项对应的向导提示步骤完成，这里不赘述。

(1) 逻辑库管理

启动 SAS 窗口环境，双击“SAS 资源管理器”窗口的“SAS 逻辑库”图标，显示该 SAS 会话可用的所有逻辑库，包括临时逻辑库 WORK、系统逻辑库 sashelp、sasuser 和用户定义的逻辑库。在“逻辑库”目录下，可以通过菜单“文件”→“新建”，或右键单击空白处从浮动菜单选择“新建”来创建新逻辑库。“新建逻辑库”对话框如图 2.7 所示。这里与通过程序语句定义 SAS 逻辑库一样，需要指定逻辑库名称、引擎和物理路径信息。还可以通过勾选“启动时启用”来指定该逻辑库在 Base SAS 启动时即创建并启用，或在“选项”输入框里指定相应选项。

右键单击刚才创建的逻辑库 saslib，会显示该逻辑库可用选项的浮动菜单。用户定义的逻辑库浮动菜单如图 2.8 所示。可以通过该浮动菜单实现：在该逻辑库中查找成员、为该逻辑库添加新的成员、删除该逻辑库和显示其属性。

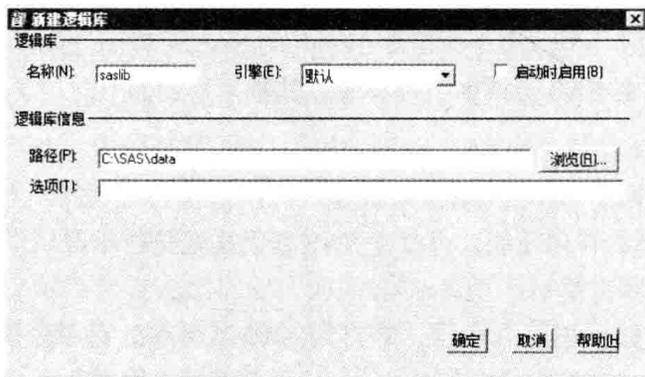


图 2.7 “新建逻辑库”对话框



图 2.8 逻辑库浮动菜单

(2) 数据集管理

通过“SAS 资源管理器”可以浏览逻辑库的内容，在浏览时，可以选定逻辑库的数据集（表）进行操作。可通过浮动菜单完成如下操作：查看数据集、将数据集导出为 Excel 文件、复制、删除、重命名 SAS 数据集和创建副本等，如图 2.9 所示。

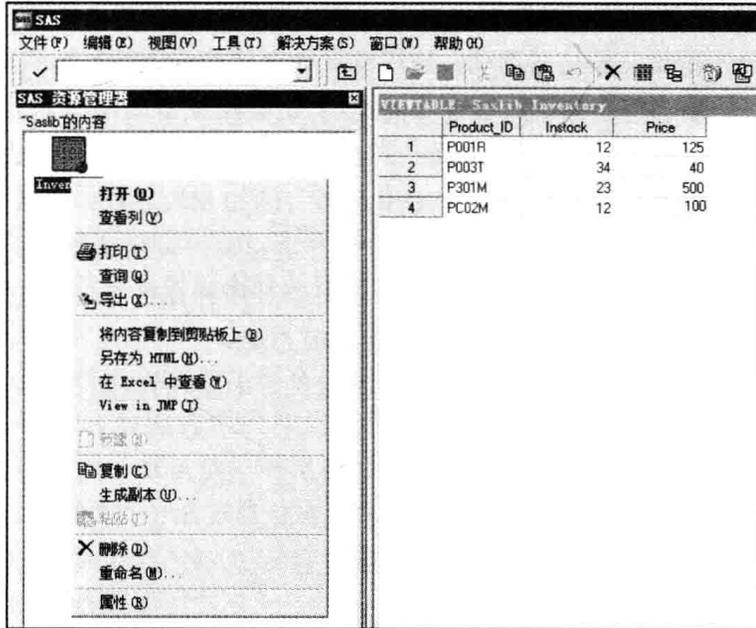


图 2.9 数据集操作

5. VIEWTABLE 窗口

在“资源管理器”窗口双击 SAS 数据集，会在 VIEWTABLE 窗口中打开当前 SAS 数据集。默认打开方式为浏览模式，该模式能保护数据不会被更改。在浏览模式下，可以定制数据集视图，例如，排序、改变列显示颜色和字体、显示标签或删除添加变量。选中数据集的一个变量（列），右键单击显示的浮动菜单如图 2.10 所示。可选择不同的菜单项，通过菜单项向导完成这些定制功能。

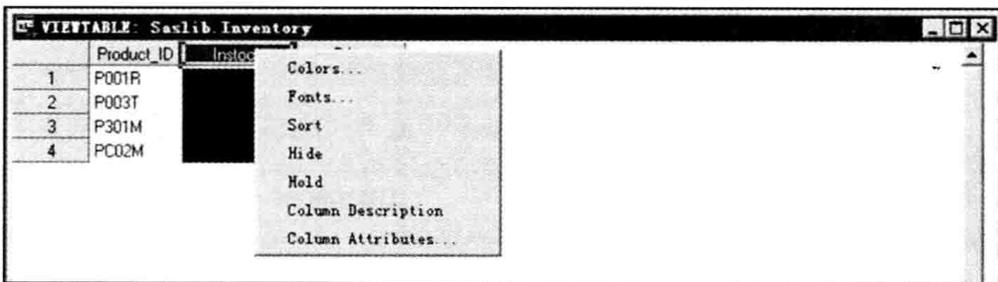


图 2.10 定制查看数据集的视图

在当前活动窗口为 VIEWTABLE 窗口时，可以通过菜单将浏览模式修改为编辑模式，选择“编辑”→“编辑模式”。在该模式下可以进行在该窗口修改数据值、按列排序等操作，并且使用菜单“文件”→“保存”或“另存为”，来保存该数据集或建立新的 SAS 数据集。

2.1.4 SAS 系统选项

SAS 的复杂之处在于存在众多选项（option），读者需要细心掌握。根据 SAS 选项出现的

位置、功能和作用范围来区分，一般分为系统选项、数据集选项和语句选项（即在语句中出现的选项）。本书会不断地介绍这些选项，建议读者针对这些选项的类别和用法进行总结。

SAS 系统选项是影响整个会话过程中 SAS 程序处理或交互式 SAS 会话的指令。SAS 系统选项所控制的内容包括 SAS 输出的外观、SAS 对所使用文件的处理形式、SAS 数据集中观测的处理形式（例如 OBS 选项），SAS 初始化特性（例如 MEMSIZE 选项），以及 SAS 如何与主机操作系统交互等。系统选项被指定时即开始产生影响，直到其被改变。

数据集选项是为数据集操作指定的选项，应用于其所作用的 SAS 数据集。有些数据集选项有对应的系统选项或 LIBNAME 选项，例如选项 OBS=。

在下面的代码中，前一个 PROC 过程受数据集选项 OBS 的控制，而后一个 PROC 过程受系统选项 OBS 的控制。

```
options `obs=10;

title "数据集选项 OBS= 生效打印 5 条观测 ";
proc print data=sashelp.shoes (obs=5);
run;

title "系统选项 OBS= 生效打印 10 条观测 ";
proc print data=sashelp.shoes;
run;
```

两个 PRINT 过程分别打印 shoes 数据集的前 5 条和 10 条观测，结果如图 2.11 所示。

语句选项出现在 SAS 语句中，用于控制该语句的行为。全局语句中的选项会有更加广泛的影响，例如 LIBNAME= 语句选项影响特定逻辑库的所有执行。如果 LIBNAME 语句的 ACCESS 选项值为 ONLY，则对该逻辑库中的所有数据集都不能进行更新或写入操作。

1. 指定 SAS 系统选项

SAS 系统选项可通过多种方式指定。常见的指定 SAS 系统选项的方法有：通过启动 SAS 时的命令行和配置文件指定，或者 SAS 启动后通过 OPTIONS 语句或 SAS 系统选项窗口指定。

在 UNIX 环境下，如果启动 SAS 时通过命令行将临时逻辑库 WORK 的物理路径设置为 /SASWORK，则 SAS 的启动命令行如下：

数据集选项 OBS=生效打印5条观测

Obs	Region	Product	Subsidiary	Stores	Sales	Inventory	Returns
1	Africa	Boot	Addis Ababa	12	\$29,761	\$191,821	\$769
2	Africa	Men's Casual	Addis Ababa	4	\$67,242	\$118,036	\$2,284
3	Africa	Men's Dress	Addis Ababa	7	\$76,793	\$136,273	\$2,433
4	Africa	Sandal	Addis Ababa	10	\$62,819	\$204,284	\$1,861
5	Africa	Slipper	Addis Ababa	14	\$68,641	\$279,795	\$1,771

系统选项 OBS=生效打印10条观测

Obs	Region	Product	Subsidiary	Stores	Sales	Inventory	Returns
1	Africa	Boot	Addis Ababa	12	\$29,761	\$191,821	\$769
2	Africa	Men's Casual	Addis Ababa	4	\$67,242	\$118,036	\$2,284
3	Africa	Men's Dress	Addis Ababa	7	\$76,793	\$136,273	\$2,433
4	Africa	Sandal	Addis Ababa	10	\$62,819	\$204,284	\$1,861
5	Africa	Slipper	Addis Ababa	14	\$68,641	\$279,795	\$1,771
6	Africa	Sport Shoe	Addis Ababa	4	\$1,690	\$16,634	\$79
7	Africa	Women's Casual	Addis Ababa	2	\$51,541	\$98,641	\$940
8	Africa	Women's Dress	Addis Ababa	12	\$108,942	\$311,017	\$3,233
9	Africa	Boot	Algiers	21	\$21,297	\$73,737	\$710
10	Africa	Men's Casual	Algiers	4	\$63,206	\$100,982	\$2,221

图 2.11 数据集选项和系统选项

```
#/opt/SASHome/SASFoundation/9.4/sas -work /SASWORK
```

如果是通过配置文件指定的，则将该选项写入配置文件中。SAS 配置文件在第 1 章中已经介绍过了。SAS 启动后在 OPTIONS 语句中指定系统选项的形式如下：

```
OPTIONS 选项 1 <选项 2> <选项 3> ...;
```

前面已经给出了通过 OPTIONS 语句给出指定 OBS= 选项值的示例。

关于通过 SAS 系统选项窗口指定和查看系统选项值的情况，会在后面介绍。

2. 查看系统选项值

如果数据集选项和语句选项出现在当前起作用的位置，可以很容易地查看。但是，要看到当前起作用的 SAS 系统选项值和当前值是通过何种方式设置的就没那么容易了，因为 SAS 为系统选项提供了默认值，并且有多种方式可以设定系统选项值。对此，可以使用 OPTIONS 过程、GETOPTIONS 函数或通过“SAS 系统选项”窗口指定选项名称来查看。

带 VALUE 选项的 OPTIONS 过程将指定选项的值、范围及该值如何设置的信息打印到“日志”窗口的基本形式如下：

```
PROC OPTIONS OPTION= 选项名称 VALUE;
RUN;
```

在 SAS 窗口提交如下代码：

```
options obs=20;

proc options option=obs value;
run;
```

在“日志”窗口打印的输出信息如图 2.12 所示。从该图可以看出，OPTION 过程打印选项 obs 值为 20，显示该值是通过 OPTIONS 语句设置的。

不使用选项 VALUE 时仅返回该选项值。

将 GETOPTION 函数作为 %SYSFUNC 宏函数的参数，从而获取系统选项设置的基本形式如下：

```
%PUT %SYSFUNC(GETOPTION(选项名称));
```

在 SAS 窗口提交如下代码：

```
options obs=20;

%put %sysfunc(getoption(obs));
```

在“日志”窗口打印输出的信息如图 2.13 所示。从该图可以看出，选项 OBS 的值为 20。

```

日志 - (无标题)
2718 options obs=20;
2719
2720 proc options option=obs value;
2721 run;

SAS (R) PROPRIETARY SOFTWARE RELEASE 9.4 TS1M0

SAS 选项 OBS 的选项值信息
值: 20
范围: DMS Process
选项值如何设置: Options Statement

NOTE: "PROCEDURE OPTIONS" 所用时间(总处理时间):
实际时间 0.04 秒
CPU 时间 0.00 秒

```

图 2.12 OPTIONS 过程打印的选项信息

```

日志 - (无标题)
62 options obs =20;
63
64 %put %sysfunc(getoption(obs));
20

```

图 2.13 GETOPTION 函数的选项信息

在 GETOPTIONS 函数中还可添加其他参数，显示指定选项的其他信息，例如默认值、启动时的值，以及该值如何设置等。

3. SAS 系统选项窗口

通过“SAS 系统选项”窗口菜单可以查看和管理 SAS 系统选项。选择菜单“工具”→“选项”→“系统”，会打开“SAS 系统选项”窗口。该窗口对 SAS 系统选项进行了分组，如图 2.14 所示。可以通过“选项组”的浮动菜单展开该分组或在该分组中查找系统选项。展开选项组、子选项组直到右侧窗口出现具体的系统选项，可通过具体选项的浮动菜单修改该选项值或将该选项值重置为默认值。

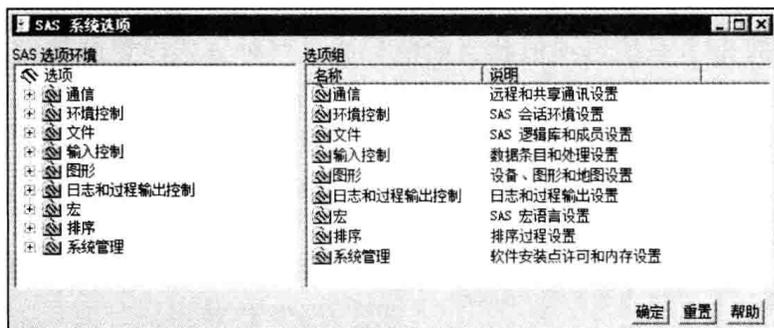


图 2.14 “SAS 系统选项”窗口

在左侧窗口单击“日志和过程输出控制”→“过程输出”，右侧窗口会显示“过程输出组的选项”窗格，如图 2.15 所示。右键单击要修改其值的系统选项，通过浮动菜单修改该选项值或将该选项值重置为默认值。例如，右键单击 Center，选择“修改值”，在“修改值”对话框选择或设置新值，单击“确定”按钮保存该选项值。

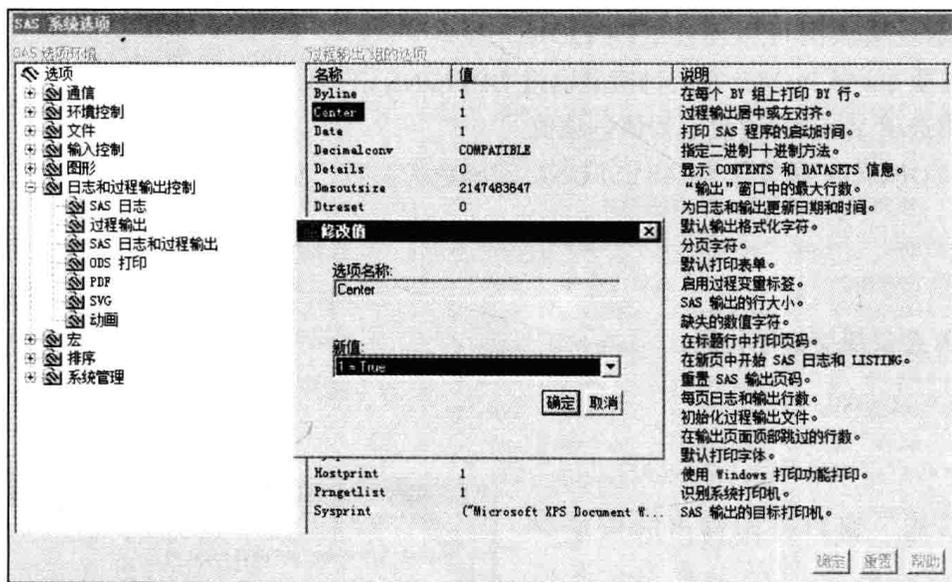


图 2.15 在“SAS 系统选项”窗口修改系统选项值

2.1.5 SAS 程序结构

SAS 程序用于访问、管理、分析和展现数据。其基础组成部分是 DATA 步和 PROC 步，PROC 步又称为 SAS 过程。一个 SAS 程序可包含以任意顺序组合的多个 DATA 步和多个 PROC 步。

DATA 步通常用于创建和操作数据集，还可用于产生定制的报表。例如，DATA 步可用于计算值、检查并修正数据中的错误、将数据存储到 SAS 数据集中以便于下次使用，以及通过对存在的数据集取子集、合并或更新，产生新的数据集。DATA 步由关键字 DATA 开始。

PROC 步是一些预先写好的例程，不同的 PROC 步其功能不同。PROC 步能够用来分析和处理 SAS 数据集中的数据，并以适当的形式展现数据和信息。有些 PROC 步会创建包含该过程结果的新 SAS 数据集。PROC 步可列出、排序和汇总数据，也可以产生描述性的统计量，并对其进行分析和优化，从而创建汇总报告、产生图表等。PROC 步由关键字 PROC 开始。

SAS 程序还包含 SAS 语句，每条 SAS 语句通常以 SAS 的关键字开始，并总是以分号结束。DATA 步和 PROC 步通常包含多条语句。SAS 语句的形式很自由，可以在一行的任何地方开始和结束，每条语句可跨越多行，多条语句也可以在同一行。语句中的“词”以空格或特殊字符分开。SAS 语句不区分大小写，但是在大多数时候，在引号中的文本是区分大小写的。

下面通过示例来理解 DATA 步、PROC 步和 SAS 语句。在图 2.16 中，SAS 语句、DATA 步和 PROC 步都已经标识出来。

```

libname saslib 'c:\sas\data'; —— SAS 语句

data saslib.Inventory; —— SAS 语句
  input Product_ID $ Instock Price; —— SAS 语句
  datalines; —— SAS 语句
P001R 12 125.00
P003T 34 40.00
P301M 23 500.00
PC02M 12 100.00
;
run; —— SAS 语句

proc print data=saslib.Inventory noobs; —— SAS 语句
run; —— SAS 语句

```

图 2.16 展示了 SAS 程序的代码片段，并用大括号和线条标注了语句和步骤。在 DATA 步中，`libname`、`data`、`input`、`datalines`、`run` 和 `;` 语句均被标注为 SAS 语句。而 `data saslib.Inventory;` 到 `run;` 之间的整个代码块被标注为 DATA 步。在 PROC 步中，`proc print` 和 `run;` 语句均被标注为 SAS 语句，而 `proc print data=saslib.Inventory noobs;` 到 `run;` 之间的整个代码块被标注为 PROC 步。

图 2.16 SAS 语句、DATA 步、PROC 步

可以在 SAS 程序的任何地方使用注释语句来说明程序的目的、解释不好理解的程序片段，或者描述复杂程序中的一些步骤或计算原理。注释有两种基本形式，第一种如下：

* 消息；

消息为注释的内容，可以是任意长度，但必须写为单独的语句，以分号结束，且内部不

9) 返回 DATA 步开始下一个迭代, 从第 3 步开始。

1. 编译阶段

当提交 DATA 步执行时, SAS 检查 SAS 语句的语法并编译它们, 将语句自动翻译为机器代码。SAS 进一步处理代码并创建输入缓冲区、PDV 和数据集描述信息。

- 输入缓冲区是内存中的逻辑区域。当程序执行时, SAS 会将原始数据文件中的每条数据记录读入该区域。
- PDV 也是内存中的逻辑区域, SAS 在该逻辑区域中构建数据集, 每次一个观测。当程序执行时, SAS 从输入缓冲区读取数据值, 或使用 SAS 语句创建数据值。SAS 把数据值赋给变量, 然后从 PDV 中将这此数据值写入为 SAS 数据集中的—个观测。
- PDV 中还包含两个自动变量 `_N_` 和 `_ERROR_`。`_N_` 变量计算 DATA 步迭代的次数, `_ERROR_` 变量作为在执行过程中由数据引起的错误的信号, 0 表示没有错误, 1 表示有一个或多个错误。这些自动变量不会写入输出数据集中。
- 描述信息是关于每个 SAS 数据集的信息, 前面介绍过。

2. 执行阶段

DATA 步中的所有可执行语句在每次迭代中都会被执行—遍。如果输入文件包含原始数据, 那么 SAS 会先读入—个观测到输入缓冲区, 然后读取输入缓冲区中的数据值并将其赋给 PDV 中的变量。SAS 还会计算由程序语句创建的数据值, 并将这些值写入 PDV。当程序执行到 DATA 步结束时, 默认会执行如下操作:

- 1) 把 PDV 中的当前观测写入数据集。
- 2) 程序循环回到 DATA 步最开始的地方。
- 3) PDV 中的变量重置为缺失值。注意, 在 RETAIN 语句中指定的变量和自动变量 `_N_`、`_ERROR_` 不会重置为缺失值。关于 RETAIN 语句会在后面的章节介绍。

如果还有另一条记录要读取, 那么程序会再执行—遍, SAS 会构建第二个观测并继续, 直到没有任何记录。这时, 数据集关闭, SAS 继续下一个 DATA 或 PROC 步。

下面通过示例来介绍 DATA 步编译和执行各条语句时输入缓冲区和 PDV 中的内容。该示例在之前用过的创建产品库存的 DATA 步基础上进行了简单的修改, 新增加了变量 Cost。代码如下:

```
libname saslib "c:\sas\data";

data saslib.inventory;
    input Product_ID $ Instock Price;
    Cost=Price*0.15;
    datalines;
P001R 12 125.00
P003T 34 40.00
P301M 23 500.00
PC02M 12 100.00
;
run;
```

- ❑ DATA 步由 DATA 语句开始，该 DATA 语句还创建命名为 Inventory 的数据集，并将其存储在 saslib 逻辑库中。
- ❑ INPUT 语句创建了 3 个变量：Product_ID、Instock 和 Price。
- ❑ 赋值语句创建了额外的变量 Cost，表示库存成本为其价格 (Price) 的 15%。
- ❑ DATALINES 语句标识输入数据的开始，数据值后的分号表示输入数据的结束。
- ❑ RUN 语句表示 DATA 步的结束。

下面借助每一步输入缓冲区和 PDV 中的数据来理解提交该 DATA 步后编译和执行阶段 SAS 的行为。

1) 提交 DATA 步执行时，SAS 首先编译 DATA 步。编译时，SAS 创建输入缓冲区、PDV 和数据集 saslib.Inventory 的描述信息。PDV 包含在 INPUT 语句中的变量、赋值语句中的变量 Cost，以及自动产生的变量 _N_ 和 _ERROR_。所有的变量，除了 _N_ 和 _ERROR_ 以外，都会初始化为缺失值。数字变量的缺失值由点 (.) 表示，字符变量的缺失值由空格表示。此时的输入缓冲区和 PDV 中的内容如图 2.17 所示。

输入缓冲区:

--

PDV:

N	_ERROR_	Product_ID	Instock	Price	Cost
1	0		.	.	.

图 2.17 输入缓冲区和 PDV 中的内容 (1)

2) 语法正确，DATA 步开始执行。INPUT 语句会让 SAS 读入第一个数据，并记录到输入缓冲区。此时的输入缓冲区和 PDV 中的数据如图 2.18 所示。

输入缓冲区:

P001R 12 125.00

PDV:

N	_ERROR_	Product_ID	Instock	Price	Cost
1	0		.	.	.

图 2.18 输入缓冲区和 PDV 中的数据 (2)

3) 根据 INPUT 语句中的指令，SAS 将输入缓冲区的值赋值给 PDV 中的变量。此时的输入缓冲区和 PDV 中的数据如图 2.19 所示。

输入缓冲区:

P001R 12 125.00

PDV:

N	_ERROR_	Product_ID	Instock	Price	Cost
1	0	P001R	12	125	.

图 2.19 输入缓冲区和 PDV 中的数据 (3)

4) SAS 执行程序中的下一条赋值语句:

```
Cost=Price*0.15;
```

该赋值语句计算变量 Cost 的值, 并将该值写入 PDV。此时的输入缓冲区和 PDV 中的数据如图 2.20 所示。

输入缓冲区:

P001R 12 125.00

PDV:

N	_ERROR_	Product_ID	Instock	Price	Cost
1	0	P001R	12	125	18.75

图 2.20 输入缓冲区和 PDV 中的数据 (4)

5) 这时, DATA 步的这一次迭代结束, 程序自动做如下操作:

- 将 PDV 中的内容作为第一个观测写入数据集, 记住, 自动变量 `_N_` 和 `_ERROR_` 不会被写入。
- 循环到 DATA 步最开始处, 开始下一个迭代。
- 释放输入缓冲区的记录。
- 在 PDV 中, 将自动变量 `_N_` 加 1, 重置自动变量 `_ERROR_` 为 0, 并将其他变量设置为缺失值。

以上操作完成后, 输入缓冲区和 PDV 中的数据如图 2.21 所示。

输入缓冲区:

--

PDV:

N	_ERROR_	Product_ID	Instock	Price	Cost
2	0

图 2.21 输入缓冲区和 PDV 中的数据 (5)

6) 继续执行。INPUT 语句查找下一条观测。在这个例子中, 存在下一条观测, INPUT 语句将第二条观测读入到输入缓冲区。此时, 输入缓冲区和 PDV 中的数据如图 2.22 所示。

输入缓冲区:

P003T 34 40.00

PDV:

N	_ERROR_	Product_ID	Instock	Price	Cost
2	0

图 2.22 输入缓冲区和 PDV 中的数据 (6)

7) 接下来 SAS 像构建上一条观测值一样在 PDV 中构建下一条观测值, 并将 PDV 的内

容写入数据集。SAS 执行完赋值语句后，输入缓冲区和 PDV 中的数据如图 2.23 所示。

输入缓冲区:					
P003T 34 40.00					
PDV:					
<u>_N_</u>	<u>_ERROR_</u>	<u>Product_ID</u>	<u>Instock</u>	<u>Price</u>	<u>Cost</u>
2	0	P003T	34	40	6

图 2.23 输入缓冲区和 PDV 中的数据 (7)

- 8) 整个过程继续，直到没有其他观测。DATA 步迭代次数与读取的原始数据观测条数一样。
- 9) SAS 关闭数据集 Inventory，本 DATA 步执行结束，退出该 DATA 步。

2.2.2 读取外部文本文件中的数据 (初级)

在使用 DATA 步读取外部文本文件中的数据时，需要给 SAS 提供读取原始数据所要求的特定信息，例如原始数据的位置、数据集变量和类型，以及 SAS 如何读数据等。SAS 会根据这些信息创建数据集。

在 DATA 步中可以使用 DATALINES 直接输入数据，或通过 INFILE 语句指定原始数据文件。本书之前的示例代码所给出的都是通过 DATALINES 直接输入数据。但在实际应用中，要分析的原始数据往往存储在外部文件中。本节主要介绍使用 DATA 步读取外部文本数据文件中的数据。

在读取原始数据文件并创建 SAS 数据集之前，必须先检查原始数据文件，确定要读取的数据值在原始数据记录中的格式，并基于这些信息选择读取数据时使用的方法。SAS 提供了以下 3 种基本输入方式：

- 列表输入
- 按列输入
- 格式化输入

这 3 种输入方式可以单独使用，也可组合使用，还可以和 SAS 提供的各种修饰符以及指针控制结合使用，例如与输入格式、行控制符、行指针控制和列指针控制一起使用。

使用 DATA 步读取外部文件中数据的基本形式如下：

```
DATA 数据集名称 ;
    INFILE 数据文件位置 ;
    INPUT 变量列表 ;
RUN;
```

其中：

- DATA 语句指定数据集名称。
- INFILE 语句指定原始数据的位置和名称。原始数据文件可以是在 FILENAME 语句中定义的文件引用形式或操作系统下的文件路径。

□ INPUT 语句用于告诉 SAS 如何读取数据。

这里简单介绍如何使用 FILENAME 定义文件引用。在 INFILE 语句中可直接使用带路径的外部文件名称，也可以使用事先由 FILENAME 语句赋值的文件引用。使用 FILENAME 语句可以将 SAS 文件引用与外部文件或者存储位置（存储有多个外部文件）关联起来，其基本形式如下：

```
FILENAME 文件引用 外部文件 | 外部文件存储位置；
```

其中文件引用用于指定文件引用的名称，以字母开始，可包含字母、数字和下划线，长度不能超过 8 个字符。外部文件则给出了一个带完整路径名的外部数据文件的文件名，或者一组文件的存储位置。下面分别给出在 Windows 环境下，FILENAME 语句指定单个文件和一组文件的存储位置定义，以及在 INPUT 语句中如何使用这两种的文件引用。

1) FILENAME 语句指定到单个文件的文件引用。

```
filename invtfile 'c:\sas\data\inventory.dat';

data saslib.Inventory;
  infile invtfile;
  input Product_ID $ Instock Price;
run;
```

2) FILENAME 语句指定到一组外部文件存储位置的文件引用。

```
filename extfiles 'c:\sas\data';

data saslib.Inventory;
  infile extfiles(inventory);
  input Product_ID $ Instock Price;
run;
```

本章后面的例子都使用后面这种方式引用外部数据文件。

1. 列表输入

列表输入（List Input）用于读取原始数据记录中每个字段由至少一个分隔符隔开，并且数据值中不包含该分隔符的原始数据。列表输入默认分隔符为空格，连续的分隔符会当成一个分隔符处理，INPUT 语句中包含了简单的变量名称列表。使用列表输入的基本形式如下：

```
DATA 数据集；
  INFILE 文件引用；
  INPUT 变量 1 <$> <变量 2 <$> ...>;
RUN;
```

其中：

- 数据集指定要生成的数据集。
- 文件引用指定要读入外部原始数据文件。

□ 变量 1、变量 2 等是数据集的变量，变量与变量之间用空格分隔。对于字符变量则需在变量后加字符 \$。INPUT 语句顺序地将原始数据记录中由空格隔开的数值读入到所列出的变量中。

(1) 原始数据各个字段以空格隔开

当原始数据记录中的数据值以一个或多个空格隔开，且数据值中不包含空格时，可使用默认的列表输入方式。

文件 inventory.dat（位于 c:\sas\data 目录下）的内容如下：

```
P001R 12 125.00
P003T 34 40.00
P301M 23 500.00
PC02M 12 100.00
```

本章所有的示例中，都假定已经为逻辑库引用名 saslib 指定了 SAS 逻辑库，而且为文件引用名 extfiles 指定了所有外部数据文件所在的位置。

在 SAS 窗口中提交如下代码：

```
data saslib.Inventory;
    infile extfiles(inventory);
    input Product_ID $ Instock Price;
run;

proc print data=saslib.Inventory;
run;
```

INPUT 语句会逐行顺序地读取 inventory.dat 中的数据值，并赋值给变量。在读取每行数据时，遇到空格就停止读入当前数据值，并从非空格处读入下一个数据值。

Product_ID	Instock	Price
P001R	12	125
P003T	34	40
P301M	23	500
PC02M	12	100

PRINT 过程打印的数据集内容如图 2.24 所示。

图 2.24 打印的数据集

上述的列表输入代码存在如下限制：

- 使用列表输入时，字符变量长度默认为 8 个字节。当输入数据长度超过 8 个字节时，从输入缓冲区读入 PDV 的数据值会产生截断。这个问题可以通过在 INPUT 语句之前使用 LENGTH 语句指定该变量的长度或其他方式来解决。
- SAS 遇到空格时会停止读入当前数据值，这样 SAS 就不能处理原始数据记录中的数据值包含空格的情况了。
- 原始数据中必须使用占位符 (.) 来表示该数据值为缺失值。

(2) 使用 LENGTH 语句指定字符变量长度

指定变量长度的 LENGTH 语句的基本形式如下：

```
LENGTH 变量1 <$>长度 <变量2 <$>长度 ...>;
```

通常在变量第一次出现时程序会根据上下文环境确定其长度。可以在 INPUT 语句前通

过 LENGTH 语句明确指定变量长度。

要读入的数据文件 state.dat 中的原始数据记录如下, 在该记录中州名全称字符长度超过了 8 个字节。

```
WA Washington
CA California
AK Alaska
AL Alabama
```

如果使用不带 LENGTH 语句的列表输入的代码如下:

```
data saslib.state;
  infile extfiles(state);
  input Short_Name $ State $;
run;

proc print data=saslib.state noobs;
run;
```

PRINT 语句打印的数据集内容如图 2.25 所示, 可以看到, 第 1 条和第 2 条观测中 State 变量的数据值不全, 都出现了截断。

可在 INPUT 语句前加 LENGTH 语句, 指定 State 变量的长度为 20 个字符。修改后的 SAS 代码如下:

```
data saslib.state;
  length State $20;
  infile extfiles(state);
  input Short_Name $ State $;
run;

proc print data=saslib.state noobs;
run;
```

Short_Name	State
WA	Washingt
CA	Californ
AK	Alaska
AL	Alabama

图 2.25 数据值被截断

这时, PRINT 打印的数据集的内容如图 2.26 所示。State 变量中的州名完全写入数据集中了。

可能已经注意到, 打印的数据集中变量的顺序发生了变化。这是因为它们的顺序由 DATA 步中变量出现的顺序决定, 而 State 变量首先在 LENGTH 语句中出现, 接着 INPUT 语句才会出现 Short_Name。但是, 这并不影响数据值的读入顺序。在列表输入方式下, 读入的数据值顺序由其在 INPUT 语句中出现的顺序确定。

State	Short_Name
Washington	WA
California	CA
Alaska	AK
Alabama	AL

图 2.26 加 LENGTH 语句后的输出

(3) 使用 INFILE 语句的选项 DLM= 指定分隔符

当原始数据中数据记录的数据值未使用空格, 而是使用其他分隔符时, 需在 INFILE 语句中使用 DLM= 选项, 告诉 SAS 读入数据时需要使用的分隔符。

下面将上面外部数据文件的内容稍作修改以便比较。文件 inventory_dlm.dat 的内容如

下，数据记录中的各数据值之间由逗号(,)隔开。

```
P001R,12,125.00
P003T,34,40.00
P301M,23,500.00
PC02M,12,100.00
```

正确读取该数据文件的代码如下：

```
data saslib.Inventory;
    infile extfiles(inventory_dlm) dlm=',';
    input Product_ID $ Instock Price;
run;

proc print data=saslib.Inventory noobs;
run;
```

PRINT 过程打印的数据集与上例一样，这里不再给出。

使用 DLM= 选项可处理原始数据记录中数据值中包含空格的情况。此外，使用 DLM= 选项的 DATA 步也可以很好地处理数据中的缺失值。如果接连有多个指定的分隔符，也会当成一个分隔符处理。但如果分隔符之间有空格，则该空格会当作缺失值读入变量并写入数据集。例如，当数据文件 inventory_missing.dat 的内容如下：

```
P001R,12,125.00
P003T,34,40.00
P301M, ,500.00
PC02M,12,100.00
```

Product_ID	Instock	Price
P001R	12	125
P003T	34	40
P301M	.	23
PC02M	12	100

提交与上例相同的 SAS 代码，记得将外部数据文件名字改为 inventory_missing.dat。PRINT 过程打印的数据集内容如图 2.27 所示，其中第 3 个观测 Instock 变量为默认值。

图 2.27 打印的数据集内容

(4) 使用 INFILE 语句的选项 DSD

指定选项 DSD 后，如果数据值是由引号引起来的，可以将数据值中的分隔符当成是数据值的一部分读入，字符值中的引号在读入 PDV 时会被删除。DSD 选项将默认的分隔符设置为逗号，还改变了使用列表输入时 SAS 处理分隔符的方式，比如，如果有两个连续的逗号，将被当作缺失值。

原始数据文件 customer_dsd.dat 的内容如下，其中客户的地址信息中包含了逗号，并使用双引号引起来，第一条和第三条记录中的客户名字缺失。

```
C001,, "14 Bridge St. San Francisco, CA"
C002,Emily Cooker, "42 Rue Marston"
C002,, "52 Rue Marston Paris"
C005,Jimmy Cruze, "Box 100 Cary, NC"
```

读取该外部数据文件的 SAS 代码如下，代码中还使用了 LENGTH 语句指定变量长度。

```

data saslib.customer;
  length Name $20 Address $40;
  infile extfiles(customer_dsd) dsd;
  input Customer_ID $ Name $ Address $;
run;

proc print data=saslib.customer noobs;
run;

```

PRINT 过程打印的数据集内容如图 2.28 所示。同样，因为 SAS 先编译 LENGTH 语句，所以数据集中 Name 和 Address 变量出现在 Customer_ID 前面了。

选项 DSD 还可以和其他选项（例如 DLM= 和 DLMSTR=）一起使用。DLM= 在上面介绍过，DLMSTR= 用于指定分隔数据值的字符串。如果需要，可以查看 SAS 帮助文档获得更多信息。

Name	Address	Customer_ID
	14 Bridge St. San Francisco, CA	C001
Emily Cooker	42 Rue Marston	C002
	52 Rue Marston Paris	C002
Jimmy Cruze	Box 100 Cary, NC	C005

图 2.28 指定 DSD 选项后的打印内容

2. 按列输入

当原始数据记录中的数据值在每条记录中占据相同的列时，可使用按列输入的方式。按列输入（Column Input）可以读取固定列的数据。该读入方式的 INPUT 语句基本形式如下：

```
INPUT 变量1 <$> 开始列 <- 结束列 > <变量2 <$> 开始列 <- 结束列 > ...;
```

其中：

- 变量名后有 \$ 表示该变量为字符型变量。
- 开始列 - 结束列指定变量在原始数据记录中所处的位置。
- 变量长度由为该变量指定的列数确定，可以超过 8 个字节。
- 按列输入可读入包含空格的数据值。
- 可以处理数据中的缺失值，不需要使用占位符。

文件 customer.dat 的内容如下，其中，第 1 ~ 14 列为产品编号，第 16 ~ 26 列为附属品牌，第 28 ~ 29 列为专卖店数，第 31 ~ 35 列为产品库存数。

```

C001                14 Bridge St.      San Francisco CA
C002 Emily Cooker   42 Rue Marston
C002                52 Rue Marston      Paris
C005 Jimmy Cruze   Box 100              Cary              NC

```

读入该原始文件中数据的 SAS 代码如下：

```

data saslib.customer;
  infile extfiles(customer);
  input Customer_ID $ 1-4 Name $ 6-19 Address $ 21-37 City $ 39-51 State $ 53-54 ;
run;

proc print data=saslib.customer noobs;
run;

```

PRINT 过程打印的数据集内容如图 2.29 所示。

在列表输入方式下，读入的数据值由指定的列号确定，所以使用列表输入可以以任何顺序读入列，而且可跳过一些列、不读入到数据集中或重复读取数据记录中的数据值。

Customer_ID	Name	Address	City	State
C001		14 Bridge St.	San Francisco	CA
C002	Emily Cooker	42 Rue Marston		
C002		52 Rue Marston	Paris	
C005	Jimmy Cruze	Box 100	Cary	NC

图 2.29 按列输入后的打印内容

3. 格式化输入

上面介绍的按列输入与列表输入一样，只能读取标准的字符或数字值到数据集中。其实，SAS 还可以读取特殊格式的数字数据，例如二进制数据、日期/时间（01FEB2013），或者包含逗号（1,262）、货币符号（\$87.3）等特殊字符的数字值。在这种情况下，就需要使用格式化输入（formatted input）了，即在 INPUT 语句中提供特殊的指令，以便 SAS 正确地读取原始数据记录中的数据值。这些特殊指令称为输入格式（Informat）。格式化输入组合了按列输入特征和读取非标准化数字或字符值的能力，保证数据值可正确地原始数据记录中读入。

对单个变量，格式化输入的 INPUT 语句的基本形式如下：

```
INPUT 变量 <$> 输入格式；
```

数值型和字符型最基础的输入格式形式分别为 n. 和 \$n.，例如 10. 和 \$20.，表示从输入缓冲区的当前列控制指针处分别读取 10 列和 20 列，并赋值给对应变量。对于字符变量，输入格式还指定了变量的长度，而数字型变量的长度仍然为 8。

除了这些基础的格式外，SAS 还提供了一些格式用于读入特殊格式的数字值，例如前面提到的读入带 \$ 符号的数字或日期时间值。同时，SAS 还支持自定义格式，在本书后面的章节会介绍。

来看个示例，原始数据文件 sales.dat 的内容如下，该文件中原始数据记录包含字段依次为员工 ID、部门、销售额和上次修改日期，其中销售额和日期都不是标准数字值，需使用对应的输入格式。

```
ET001 TSG $10000      01JAN2012
ED002      $12000      01FEB2012
ET004 TSG $5000       02MAR2012
EC002 CSG $23000      01APR2012
ED004 QSG              01AUG2012
```

读入处理该文件的 SAS 代码如下，其中 Sales 和 Date 变量分别使用了输入格式 comma6. 和 date9.，Emp_ID 和 Dept 使用的是上面介绍过的按列输入方式。

```
data saslib.sales;
    infile extfiles(sales);
    input Emp_ID $1-5 Dept $7-9 +1 Sales comma6. @22 Date date9.;
run;
```

```
proc print data=saslib.sales noobs;
run;
```

PRINT 过程打印的数据集内容如图 2.30 所示。SAS 的日期、时间以数字形式存储。所以打印的数据集中 Date 变量的值为数字，其实可通过输出格式（format）输出为更加易读的形式。

SAS 提供了 commaw. 和 datew. 等很多格式，用于读取对应的带嵌入符号和日期的数字值，其中的 w 表示读入数据的总宽度，包含美元符号。输入格式将数据值中嵌入的 \$ 符号转换成了不带美元符号的数字值写入到 PDV 中，并最终写入数据集，如图 2.31 所示。

Emp_ID	Dept	Sales	Date
ET001	TSG	10000	18993
ED002		12000	19024
ET004	TSG	5000	19054
EC002	CSG	23000	19084
ED004	QSG		19206

图 2.30 格式化输入数据的打印内容

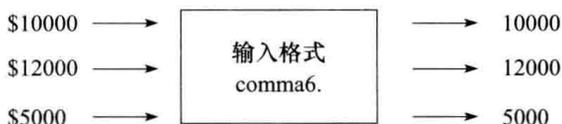


图 2.31 带格式数据的输入与输出

在 INPUT 语句中，还使用了相对列控制符号 +1 和绝对列控制符号 @22，分别表示将当前的输入列控制指针向前移 1 位和将该指针直接移动到列 22。在上面的示例中，程序读入一行记录到输入缓冲区后，列控制指针的移动情况如下：

- 第 1 ~ 5 列写入 Emp_ID，列控制指针在第 6 列。
- 第 7 ~ 9 列写入 Dept，这时列控制指针在第 10 列。
- +1 将列控制指针移到第 11 列。
- 开始读入 comma6. 中指定的 6 列，即将第 11~16 列使用输入格式转换后写入 Sales，这时列控制指针在第 17 列。
- @22 将控制指针直接移到第 22 列，读入 date9. 中指定的 9 列，即第 22~30 列，然后使用该输入格式进行转换，并写入 Date。

SAS 还提供了丰富的输入格式以便读入各种形式的数据值。下面给出了 SAS 提供的常用数字、字符、日期、时间、日期时间输入格式，如表 2.1 和表 2.2 所示。关于这些输入格式的详细使用方法，请参考 SAS 帮助文档学习。

表 2.1 常用数字和字符输入格式

格式名称	描述	示例
\$BINARYw.	将二进制数据转换为字符数据	0100110001001101
\$CHARw.	读取带空格的字符数据	\$CHAR20.
\$w.	读入标准字符数据	\$20.
BINARYw. d	将正二进制值转换为整数	10100101
COMMAw.d	读取数字值并将嵌入的字符删除	\$12,038.45
w.d	读入标准的数字数据	12.2

本书仅提供部分阅读，如需完整版，请联系QQ: 461573687

提供各种书籍pdf下载，如有需要，请联系 QQ: 461573687

PDF制作说明：

本人可以提供各种PDF电子书资料，计算机类，文学，艺术，设计，医学，理学，经济，金融，等等。质量都很清晰，而且每本100%都带书签和目录，方便读者阅读观看，只要您提供给我书的相关信息，一般我都能找到，如果您有需求，请联系我 QQ: 461573687, 或者 QQ: 2404062482。

本人已经帮助了上万人找到了他们需要的PDF，其实网上有很多PDF,大家如果在网上不到的话，可以联系我QQ。因PDF电子书都有版权，请不要随意传播，最近pdf也越来越难做了，希望大家尊重下个人劳动，谢谢！

备用QQ:2404062482