The Little SAS® Book

A Programming Approach

a primer

FOURTH EDITION

Lora D. Delwiche and *Susan J. Slaughter*

第一章 SAS 软件入门

1.1 SAS 语言

许多软件要么是菜单驱动，要么是命令驱动（输入命令——看结果）。SAS 两者都不是，在 SAS 中，你用一个叫做 SAS 程序的一系列指令语句，这些程序可以表达出你想做的事情，并用 SAS 语言写下来。SAS 有菜单驱动栏，比如 SAS 企业向导模块，它使 SAS 看起来像一个点击的软件，但这些模块仍然使用 SAS 语言为你写程序。如果你试图用 SAS 写下你自己的程序，那就要具备一定的灵活性。

SAS 程序 一个 SAS 程序就是一个按顺序执行的语句序列，一个语句给 SAS 下达信息和指令，且必须要正确的安放。一个常用来与 SAS 程序做类比的例子是去银行取款，你进入银行、排队、轮到你，那么你会对柜台谁你想做的事，叙述语句可能会是这样：

I would like to make a withdrawal.

My account number is 0937.

I would like$200.

Give me five 20s and two 50s.

注意第一句话说了你想做的事情，之后把相关信息传递给柜台并帮你完成要求。这里信息传递的顺序不重要，重要的是在你的叙述中，首先要说明你要做什么。你不能先说：“Give me five 20s and two 50s.”这会使柜台小姐一头雾水。此外，你必须确保后面的语句都围绕第一句展开。

SAS 语句 像任何语言一样，SAS 语句的编写也需要遵守一些语法规则。幸运的是，相比英语来说，SAS 语句的规则不仅少，而且简单。

最重要的规则是：

每一个 SAS 语句都由一个分号结尾

听起来很简单，但即使最富有经验的 SAS 程序员也会偶然忘记分号。如果你能记住这个规则，再来看看另外两个规则吧。

SAS 程序布局 让每一条语句看起来整洁、用缩进来表现语句的各个部分，这是很有用的，但不是必须的：

- SAS 语句不区分大小写。
- 一条语句可以持续到第二行（只要不把一个单词分开）。
- 几条语句可以用一行。

- 可以在任何一列中开始一条语句

注释 可以在你的程序中插入一些注释，让它更容易明白。即使你插入一些你喜欢的食物品名也不会对程序有所影响，因为 SAS 不读取注释。但不要忘记注释是为了让某人更轻松的学习你的程序，并明白你为什么这么做。

```
*Read animals'weights from file;

DATA animals;

INFILE'c:\MyRawData\Zoo.dat';

INPUT Lions Tigers;

PROC PRINT DATA=animals;   /*Print the results*/

RUN;
```

有两种注释方法，一种是*号和；号；一种是用/* */表示，注意第二种注释方法不能放在第一列

错误 SAS 程序通常将执行的错误标注为醒目的红色字母，你可能忘了分号，拼错了字母，按错了键盘，一个小错误会使得整个程序无法运行。当你看到红色部分多余黑色部分的时候，不要灰心。

1.2 SAS 数据集

在你进行分析、撰写报告、对你的数据进行任何处理之前，SAS 必须能够处理你的数据，你的数据必须是一种叫 SAS 数据集的特殊形式。因为 SAS 非常灵活，能够读取任何形式的数据，所以将你的数据变成 SAS 数据集是一件非常简单的事。

变量和观测值 在传统的 SAS 术语中，数据包括变量和观测值。采用相关的数据库的术语，SAS 数据集也被叫做表、观测值也被叫做行、变量也被叫做列，你可以看到下面这个包含一些数据的表。

		Variables (Also Called Columns)			
		Id	Name	Height	Weight
Observations (Also Called Rows)	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 未加工的数据有多种形式，但 SAS 将其简单化。在 SAS 中只有两种数据类型——数值型和字符型。数值型完全是数据，可以被加减乘除、可以是正负且是小数。字符变量是除数值之外的类型，可以是数值、字母、和一些特殊的字符（￥、!），最多可以占用 32767 个字节长度。

如果一个变量既包括数字又包括字符，那么它一定是字符变量。如果只包括数字，可能是字符变量也可能是数值变量。在上面这个表中，姓名是字符变量，身高和体重是数值变量，ID，既可能是数值有可能是字符，依据你的选择。

缺失值 数据有时会有些不完美，某些变量的个别观测值会缺失。字符变量的缺失值用空格表示，数值变量的缺失值用句号（.）表示。上表中，体重的第五个观测值缺失，用.表示。姓名的第六个观测值缺失，用空格表示。

SAS 数据集的大小 在 SAS 9.1 之前（prior to SAS 9.1），SAS 数据集可以包含 32767 个变量，从 SAS 9.1 开始（beginning with SAS 9.1），SAS 可包含的最多变量数由你的电脑可用资源决定（内存，CUP？）。但是超过 32767 个变量的 SAS 数据集不能用在早期的 SAS 版本上。

SAS 命名规则 为你的变量和数据集命名，使它们容易被辨别。A,B,C 这样的名字可能看起来很完美，写程序的时候也很方便，但当你 6 个月后再使用这些数据时，你会发现 name，height，weight 这样的名字更有用。为变量和数据集命名时要遵守如下规则：

- 名字的长度要小于等于 32 个字节。
- 以字母或下划线开头。
- 可以包含字母、数字、或者是下划线，不能是%$!*&#@。
- 可以是小写或大写字母，且不区分大小写。

SAS 数据集储存的文件 SAS 数据集包含了一些类似名称、创建日期、创建用的 SAS 版本等信息。SAS 也储存了每个变量的信息，包括名称、类型、长度、数据集中的位置。这些信息叫做数据集的描述部分，它使得数据集可以自我编制（self-documenting）。

1.3 SAS 程序的两个过程

SAS 程序有两个基本模块：数据步和过程步。一个典型的 SAS 程序，由数据步创建 SAS 数据集开始，再由过程步分析数据。这里有一个例子：数据步中将米转化成千米，过程步中输出结果

DATA step
```
DATA distance;
   Miles = 26.22;
   Kilometers = 1.61 * Miles;
```

PROC step
```
PROC PRINT DATA = distance;
RUN;
```

数据步和过程步由语句组成（废话），一个过程少至 1 条语句、多至几百条。新手常犯的错误是将两种过程语句用混，只要记住数据步负责读取、修改数据，过程步负责分析数据、输出报告和效用函数，就不会犯错。

数据步由 DATA 语句开始：data+数据名。上例中数据步处理了名为 distance 的数据。为了读取外部数据、未加工的数据，数据步提供了 DO LOOPS,IF-THEN/ELSE，以及一些数值和字符函数。数据步也可以按照你想要的方式合并数据集，包括联接（concatenation）和合并（match-merge）。

过程步由 proc 语句开始：proc+过程名（print、sort、means…），SAS 过程步可以处理从数据储存、输出到方差分析、3D 图表的一切操作。

当程序遭遇 DATA\PROC 等标志着新程序开始的语句时，程序结束。如果运行的是批处理，则 run 代表语句的结束。Run 告诉 SAS 去执行所有之前的程序行，全局变量不是 DATA 或 PROC 过程的部分。上图的那个程序，当 proc 出现时，代表 data 过程结束。

典型的程序是以 DATA 语句开头，输入或修改数据，然后将数据传递给 PROC 语句。但并不一定非要用这种模式来混合 data 和 proc 语句，你可以用任何顺序来排列 data 和 proc 两者的顺序，一个程序甚至可以仅有 data 语句或 proc 语句。

下表是 data 语句和 proc 语句的一些基本不同点:

DATA steps	PROC steps
▶ begin with DATA statements	▶ begin with PROC statements
▶ read and modify data	▶ perform specific analysis or function
▶ create a SAS data set	▶ produce results or report

这只是一个简化表，SAS 软件非常灵活，所以 data 语句和 proc 语句之间真正的区别也是很模糊的。记住，这个表并不是说 proc 语句永远不能创建 SAS 数据集，或者 DATA 语句永远不能够分析生成报告。

1.4 数据步的内置循环

Data 步读取并修改数据，让你以灵活的方式控制处理数据。Data 步也有一个潜在的、内置的循环语句。你不用告诉 SAS 去执行这个循环，SAS 会自动执行。

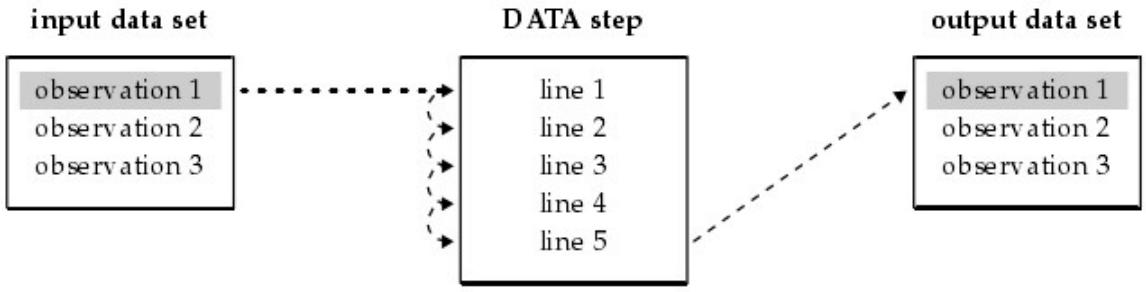
数据步按照一行一行、一个观测值一个观测值的顺序执行

这句话的表意并不明确，许多新手直到成了老手都没明白这句话的含义。

数据步“一行一行的执行”，这句话很好理解。但很多新手还是容易在这里出错，例如在没有创建一个变量之前就使用它，如果 Z 变量是 X、Y 两个变量组合的新变量，那么必须确定创建 Z 变量的语句在创建 X、Y 变量语句之后。

而“一个观测值一个观测值的执行”就不是那么容易理解。这意味着 SAS 先读取一个观测值，然后对这个观测值进行数据步的所有语句（当然也是一行一行的），然后再读取第二个观测值执行。每次执行 SAS 只有一个观测值。

我们将 SAS 执行的图景放慢：SAS 从你的数据集中读取一个观测值。SAS 对你的这个观测值执行数据步，如果数据步一直运行到结束而没有错误，SAS 会把当前的观测值写入一个新的、输出数据集中，并返回到数据步开头，读取第二个观测值进行执行。当最后一个观测值都被写入输出数据集中之后，SAS 结束数据步，进入下一个步。



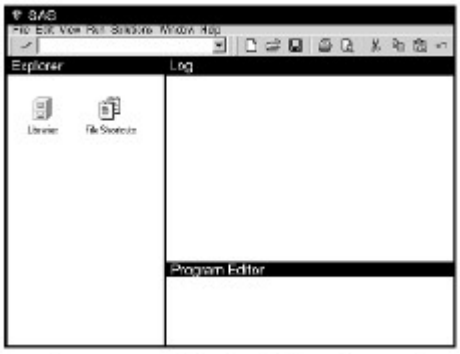
有一个类比，数据步就像是一个投票程序。当你来到投票的地点，你会站在别人后面进行排队，排到你时，你会被问到：你叫什么名字，住在哪里。当你回答之后，你可以投票。在这里，排队的人就像是观测值，投票的程序就像是数据步。一次只能让一个人投票，每个人都相互独立。并且投票的程序是一步一步来的，你不能没说明自己的姓名和住址之前就投票。

1.5 选择一个提交程序的方式

目前为止我们讨论了写 SAS 程序，但仅仅写不能带给你任何结果，你必须要提交并执行。有数种方法可以执行 SAS 程序，但不是任何方法都适合于你的操作环境。查找一下 SAS 帮助文

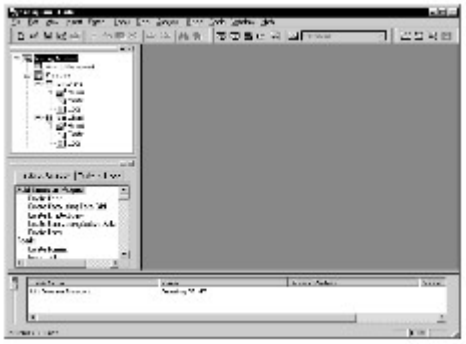
档，或者咨询下你的 SAS 顾问，看看哪种方法适合你的操作环境。

SAS 视窗环境



如果你使用SAS是按照系统提示，或者是点击SAS的图标，那么你适合使用 SAS 视窗环境。在这种交互式的环境中，你可以写入、编辑 SAS 程序，提交处理、浏览、输出结果的 SAS 程序。此外，视窗有许多功能可以处理不同的任务，如管理 SAS 文件、定制界面、访问 SAS 帮助文档、导入和导出数据。你的视窗环境的界面取决于你电脑的的类型、使用的终端、电脑操作系统和启动 SAS 时实际的选择。如果你使用的是个人电脑，那么 SAS 视窗环境的感觉和其他软件类似。

SAS 企业向导



如果你有SAS企业向导软件，这个软件在windows下即可运行。你可以用这个软件提交程序：使用插入菜单打开代码窗口，输入序或打开现有 SAS 程。之后你可以用本地电脑、或者在远程服务器上（需要安装）运行 SAS 程序。

非交互式模式

非交互式模式是 SAS 程序语句已先存于你系统的文件中，直接执行那个文件。非交互式模式可以让 SAS 立即执行程序，通过某个指令开始（$），后接文件名，如：

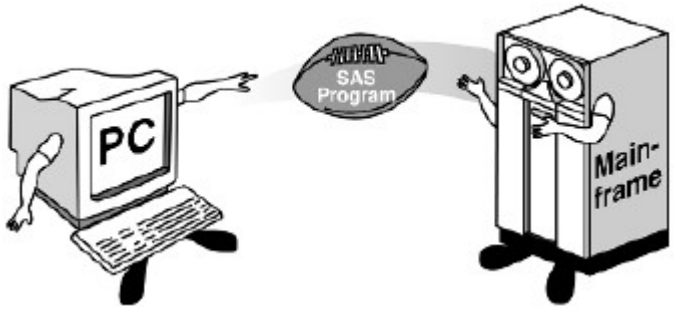
$ SAS Myfile.sas

批处理或后台模式

在批处理或后台模式下，你的程序存于一个文件中，SAS 会自动执行，你不需要在电脑旁，如果程序多，SAS 会将这个程序进行排队等待。这种模式通常用在大型电脑中，因此通常可以一次性处理多个任务。批处理或后台模式的成本比较低，适合于大型工作，工作完成后，结果会存于文件夹中，你可以任何时候输出查看。批处理未必适合你的操作环境，另外提交方式也会有不同，最好查看 SAS 帮助文档，或咨询 SAS 顾问。

远程提交

如果你安装了 CONNECT 模块，可以进行远程提交，即在一台电脑上（本地）编写程序，在另一台电脑上（远程）处理，结果会返回本地电脑。当你处理大型任务，而你的电脑性能又不够时，可以连接到远程的高性能电脑上，也可访问远程电脑的分享文件。



交互行模式

交互行模式下，SAS 每次提示用户输入一个语句，想改正输入的语句不是那么容易的。因此除非你足够优秀和熟练，否则最好不要用这个模式。你可以用 endsas 并回车来退出这个模式：

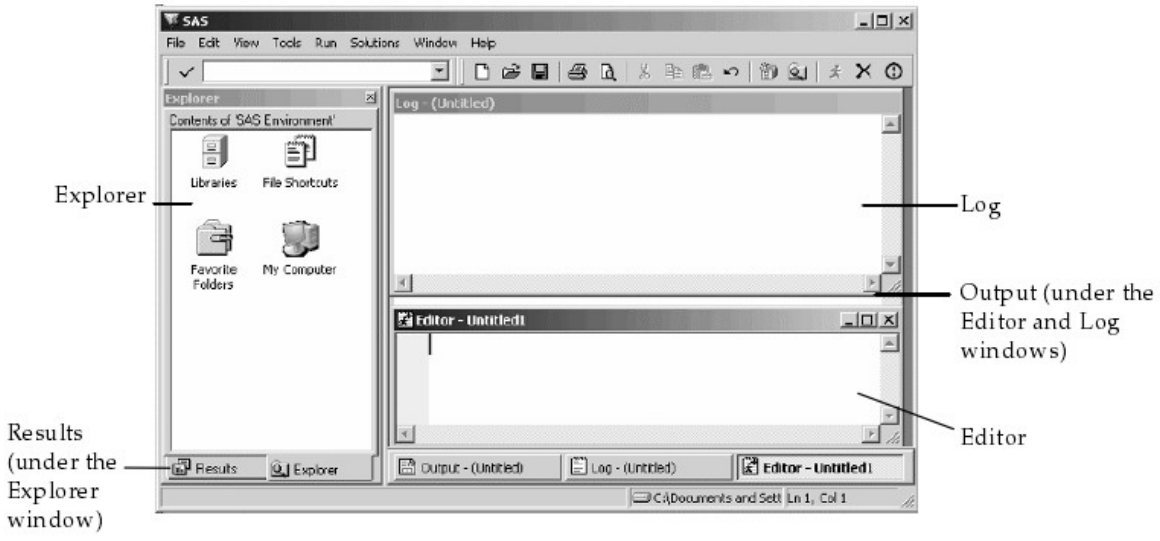
```
Endsas;
```

如果你想知道为什么会进入这个模式，并且在以后避免进入，你需要咨询 SAS 顾问。

1.6 SAS 视窗环境中的视窗和命令

SAS 视窗

SAS 有五种基本的视窗（窗口）：结果视窗、资源管理器视窗、和三种程序视窗：程序编辑、日志、输出。除此之外，在获得 SAS 帮助、改变 SAS 系统选项、定制 SAS 人机会话等情况时，可能还会用到其他的视窗，下图显示了 Microsoft Windows SAS 会话中默认的视图：



编辑窗口 编辑窗口中你可以输入、编辑、提交 SAS 程序。Windows 操作环境默认的是增强型编辑窗口，它对语法更敏感，并用颜色标注程序，使得更容易理解和发现错误。其他操作环境默认的是程序编辑窗口，并随操作环境和 SAS 版本的不同，界面特征也不同。

日志窗口 日志窗口是关于 SAS 会话的说明。在提交 SAS 程序之后，任何的说明、错误、警告和程序语句都会显示在日志窗口上。

输出窗口 如果程序产生需输出的结果，那么会反映在在输出窗口中。

结果窗口 结果窗口就像输出窗口的一个目录表，以提纲形式列出了输出的每一个部分。

SAS 命令

SAS 命令是为了不同的任务，你有三种方式发出命令：菜单、工具栏、SAS 命令栏，如下图：

菜单 （大部分操作环境都会有一个下拉菜单要么在窗口上方要么在屏幕上方…..oh my god! 略）

工具栏（不是所有的操作环境都有工具栏…略！）

SAS 命令栏 可以在这里输入 SAS 命令，一些操作环境中，命令栏坐落在工具栏中，另一些操作环境中，每个 SAS 窗口都有一个命令行（command line），通过语句 command=>激活。大部分命令是可以直接用菜单进行选择的。

控制你的视窗 你可以通过菜单、命令栏、点击的方式激活任何一种程序窗口

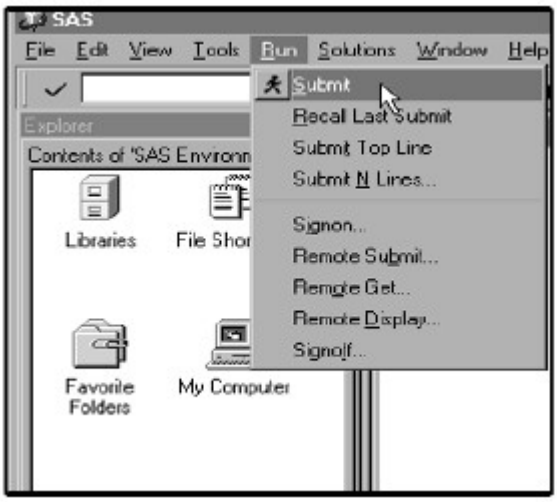
1.7 在 SAS 视窗环境中提交程序

将你的程序放入编辑窗口中 你可以通过输入，或者打开现有程序文件的方式将程序放入编辑窗口中。打开现有的程序文件，可以通过菜单-打开，也可以通过工具栏的图表，或者直接将文件拖放到编辑窗口中。

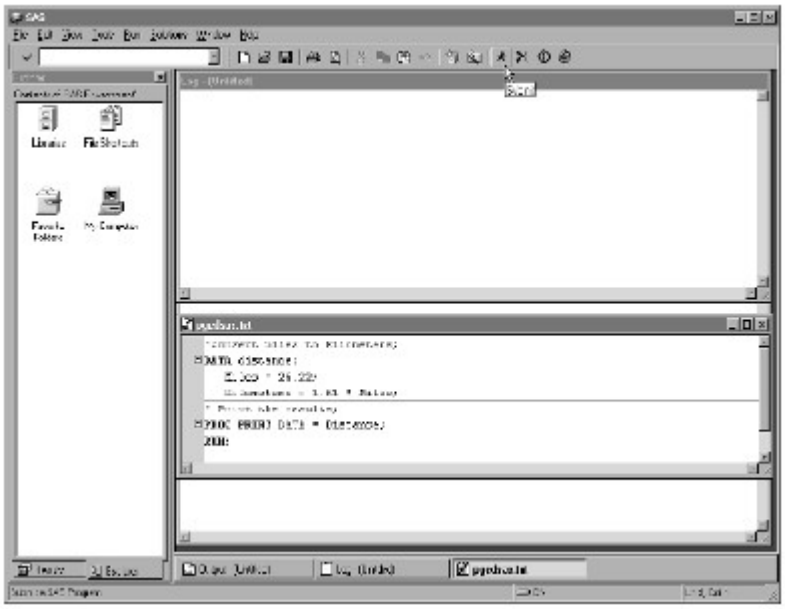
提交你的程序 你有几种方式来提交程序：

使用工具栏的提交图表

激活命令栏，输入 submit 命令回车。

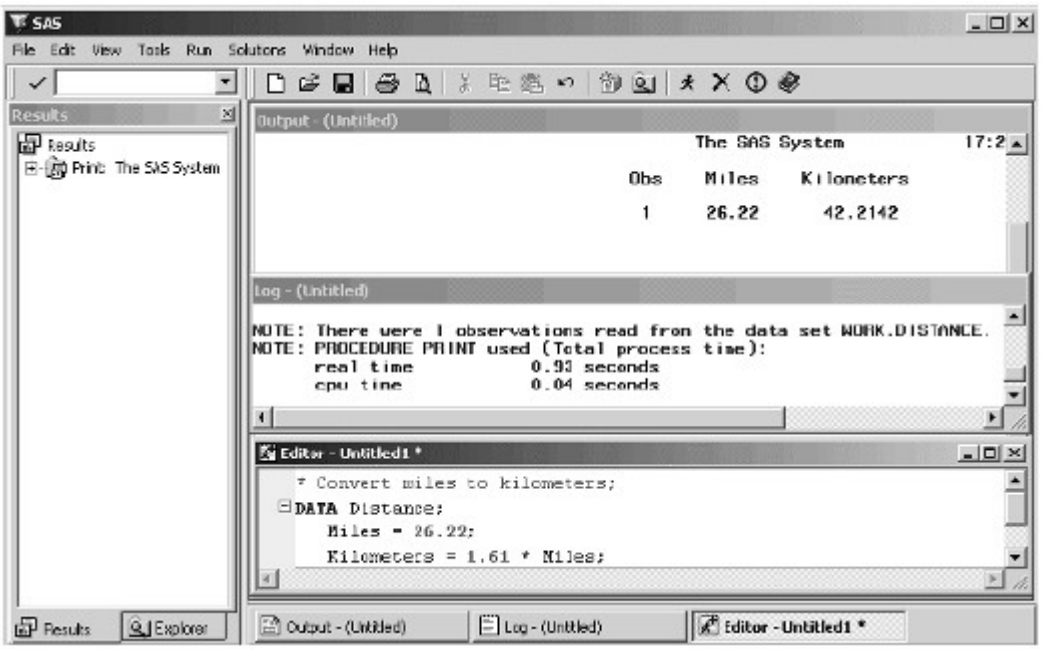


从 run 运行的下拉菜单中选择 submit 提交



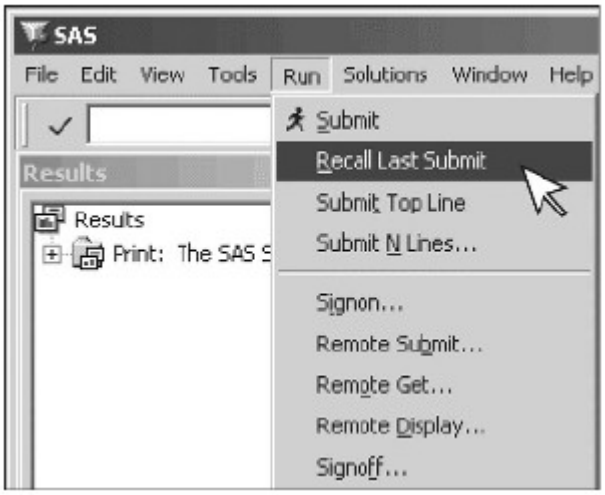
左图显示了如何在 windows 视窗中提交增强型编辑窗口的程序。

访问 SAS 日志和输出 提交程序后，日志窗口和输出窗口会有相应的日志和结果显示，如果你使用的是增强型编辑窗口，之前的程序会保留，如果使用的是程序编辑窗口，之前的程序不会保留。如果你的程序产生了输出，那么结果窗口会显示这些输出的目录，下图是一个例子，显示了提交程序之后，增强型编辑窗口、日志窗口、结果窗口、输出窗口的样式。



取回你的程序 如果不幸你的程序出现了问题，你需要再次运行，对于程序编辑窗口，由于之前的程序不在保留，因此需要调回命令（recall），有两种方法：

命令窗口中输入 recall

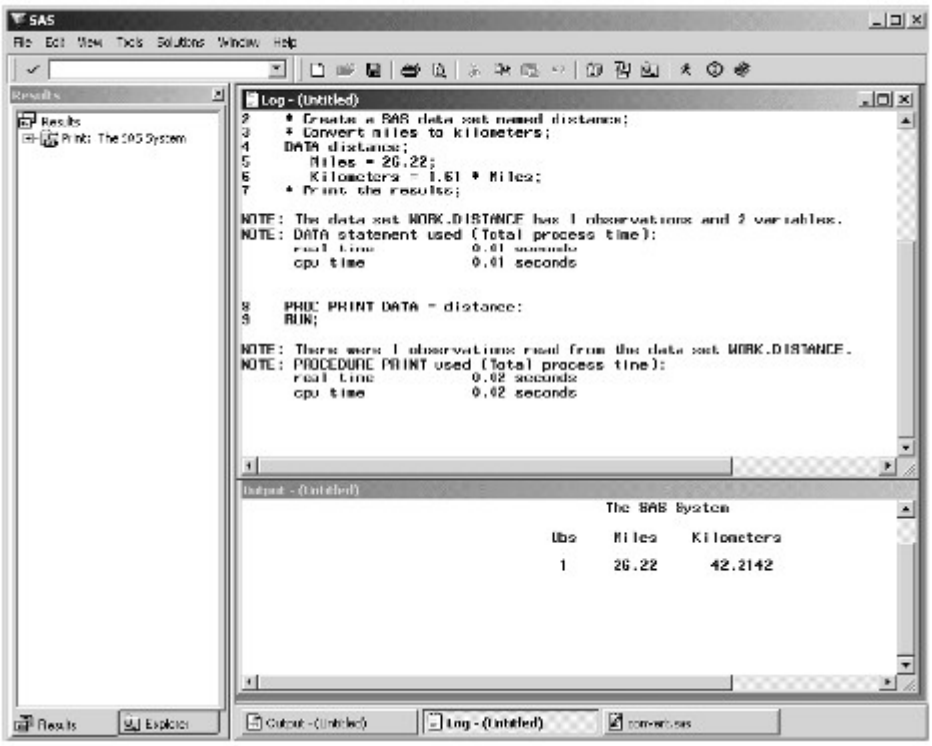


激活编辑窗口，从运行（run）下拉菜单中选择

如果不停的使用 recall 命令，SAS 可以一直往前调回程序，知道所有提交的程序都被调回。

1.8 阅读 SAS 日志

在哪找到 SAS 日志 SAS 日志窗口的位置随着你使用的操作环境、选择的模式（视窗、非交互、批处理）、个人的设置的不同而不同。在视窗模式下，提交程序之后，日志窗口默认的位置如下图：



对于批处理和非交互模式的日志则会被写入一个文件中，你需要使用操作环境的命令来查看，一般日志文件的名字与对应的 SAS 程序名一直，如你的 SAS 程序命为 abc.sas，那么日志文件的命则为 abc.log

日志包含的内容 日志中虽然有很多琐碎，但也包含了重要的信息。这里有一个将英里转换为千米的程序：

```
*Create a SAS data set named distance;

*Convert miles to kilometers;

DATA distance;

Miles=26.22;

Kilometers=1.61*Miles;

*Print the results;

PROC PRINT DATA=distance;
```

RUN;

运行之后，SAS 的日志窗口会产生一个类似这样的日志：

```
❶ NOTE: Copyright (c) 2003 by SAS Institute Inc., Cary, NC, USA.
  NOTE: SAS (r) Proprietary Software Version 9.00 (TS M0)
        Licensed to XYZ Inc., Site 0098541001.
  NOTE: This session is executing on the XP_PRO  platform.

  NOTE: SAS initialization used:
        real time             1.40 seconds
        cpu time              0.96 seconds

❷ 1     * Create a SAS data set named distance;
  2     * Convert miles to kilometers;
  3     DATA distance;
  4        Miles = 26.22;
  5        Kilometers = 1.61 * Miles;
  6     * Print the results;

❸ NOTE: The data set WORK.DISTANCE has 1 observations and 2 variables.
❹ NOTE: DATA statement used (Total process time):
        real time             0.03 seconds
        cpu time              0.03 seconds

❷ 7     PROC PRINT DATA = distance;
  8     RUN;

  NOTE: There were 1 observations read from the data set WORK.DISTANCE
❹ NOTE: PROCEDURE PRINT used (Total process time):
        real time             0.01 seconds
        cpu time              0.00 seconds
```

① 说明了你使用的 SAS 版本和 site。

② 是原始的 SAS 程序语句

③ 说明了数据步为你创建的数据集名称，观测值数和变量数。它可以帮助你确认你的程序没有丢失观测值，也没有创建你不需要的变量。

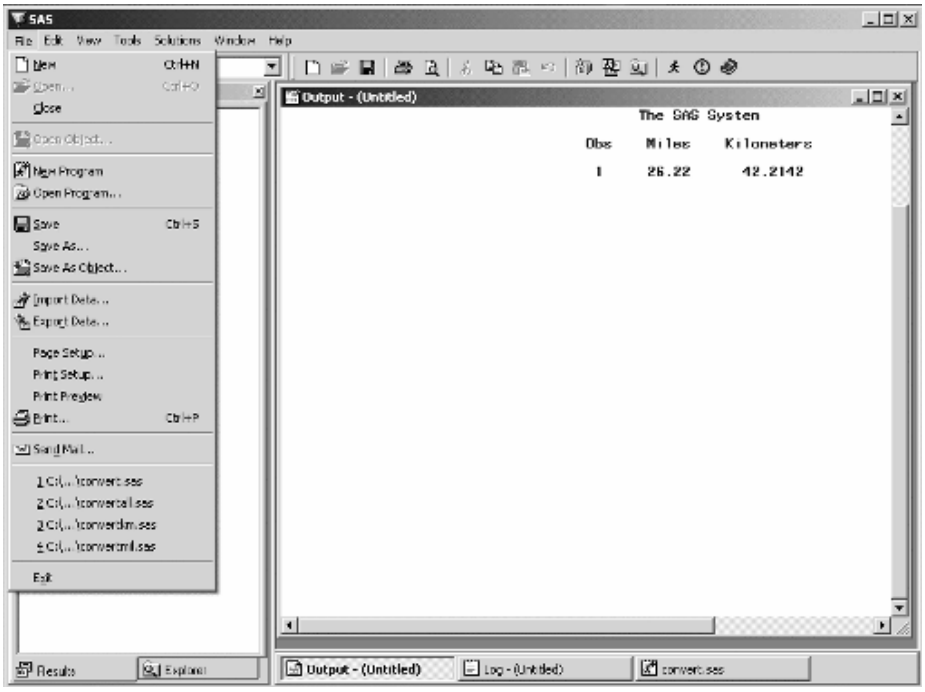
④ 这部分显示了数据步和过程步占用的电脑资源。当你使用的是多用户系统，或者处理大型数据而使得 SAS 运行占用大量时间时，这部分信息可以帮助你检查哪一步比较耗时。

如果 SAS 程序出现错误，错误信息也会反馈在日志窗口中，说明哪里出错及出错表现。

1.9 输出窗口中浏览结果

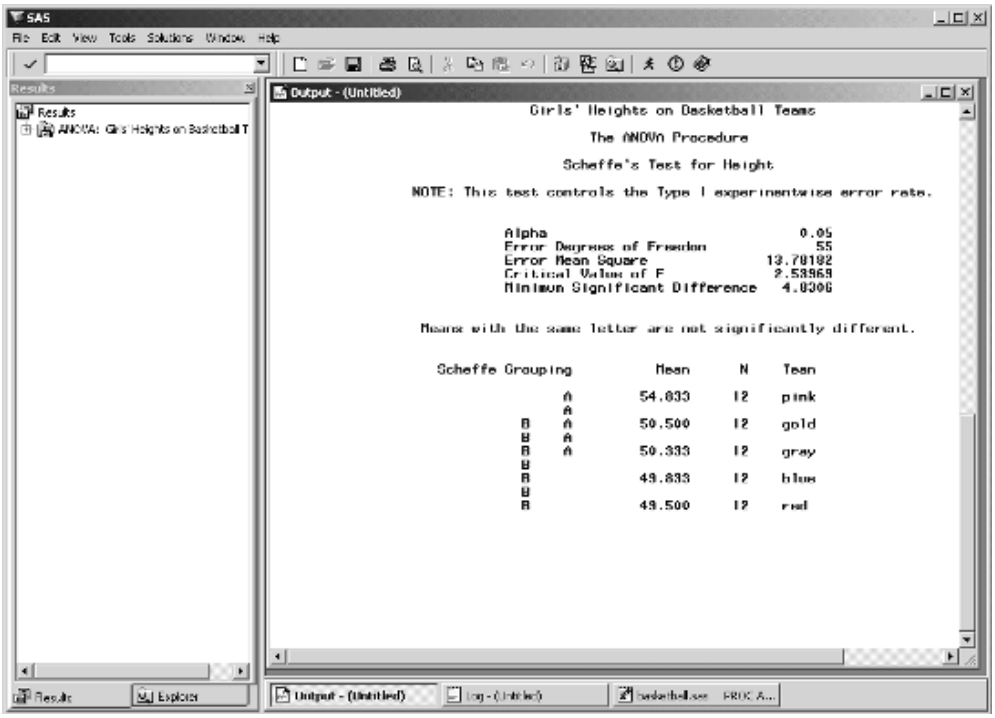
你使用的模式不同，产看输出结果的方法也会不同。如果是在视窗模式下提交 SAS 程序，那么直接在输出窗口查看，如果是批处理和非交互模式下，那么结果就会保存在一个文件里，需要用命令查看。比如使用 UNIX 系统下的非交互模式，结果会存在一个后缀为.lst 的文件里，使用 cat 或其他更多命令来查看。

输出窗口 提交程序后，结果会出现在输出窗口中，下图是一个输出窗口的例子

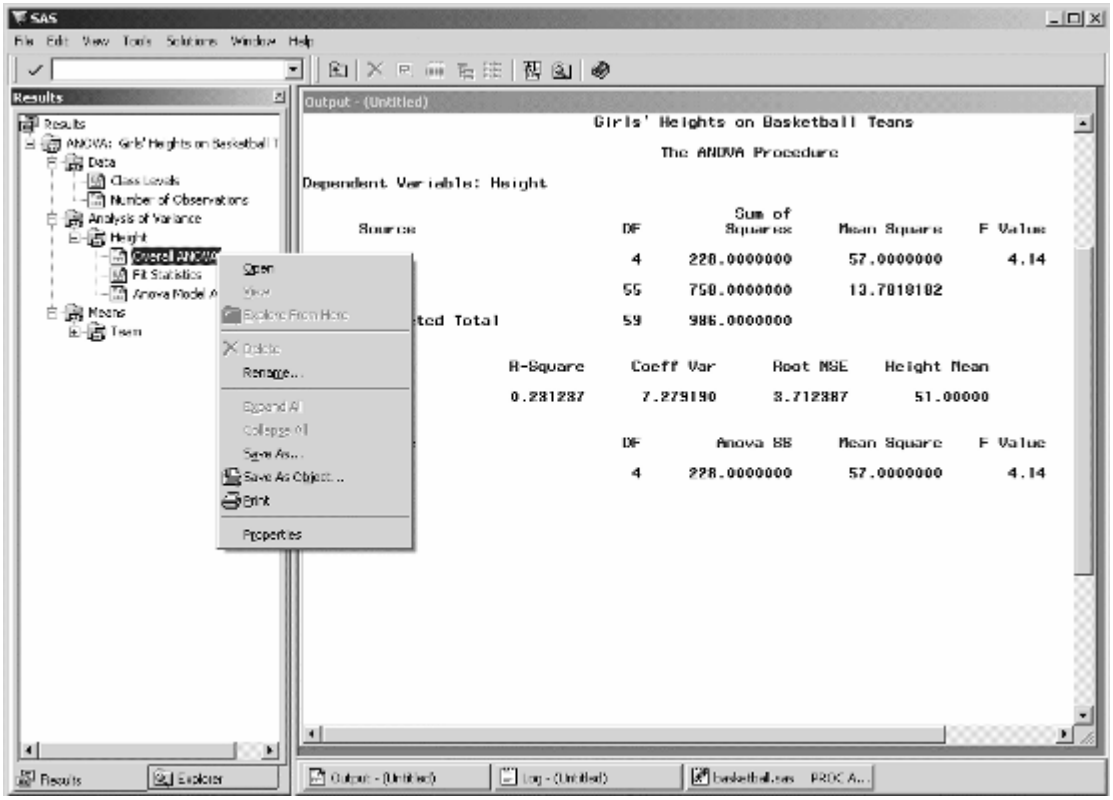


打印或保存输出窗口的目录 激活输出窗口的目录，在菜单栏文件（file）下拉菜单中选择打印（print）或保存（save）即可。

结果窗口 结果窗口起到输出窗口的目录作用，当你的输出结果非常多时，结果窗口很有用。它可以让你很清楚的看到输出结果的每一部分。下图是一个方差分析（ANOVA）过程的结果。在这个图中，左边是输出结果的目录，点击+号将其展开，可以看见 ANOVA 结果的各个部分，双击某一部分，则该部分结果就会位于输出窗口顶部。



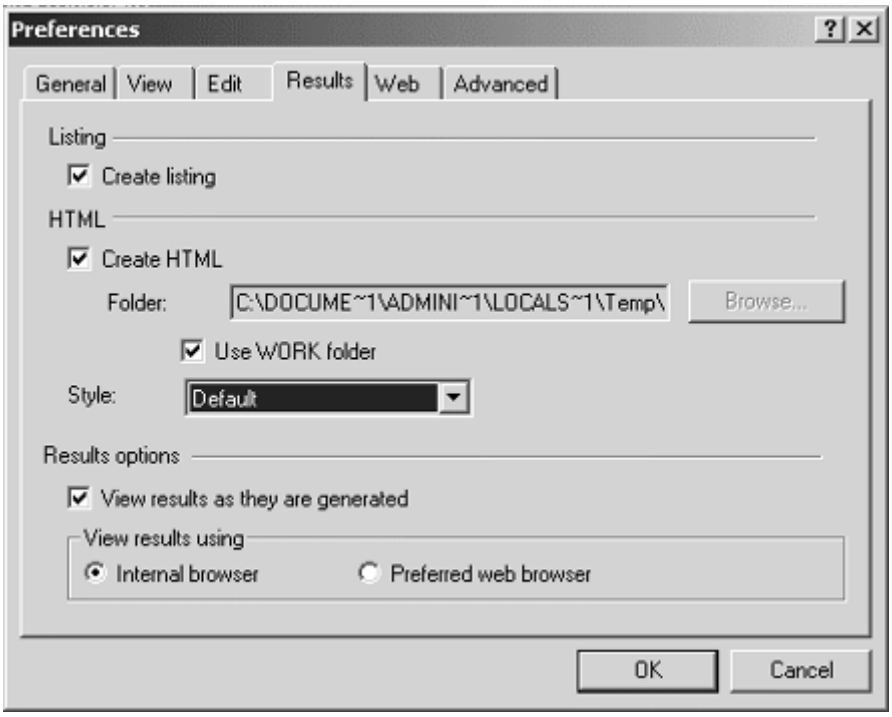
打印或保存部分输出 如果要打印结果窗口中显示的某一部分的输出，则需将鼠标移到该部分上，右击，选择打印或保存即可。或者点击一下，使其黑亮，再从菜单栏文件（file）的下拉菜单中选择输出或保存。



1.10 创建 HTML 输出

如果使用的是 SAS 视窗环境，那么可以为结果创建超文本标记语言（HTML）格式。

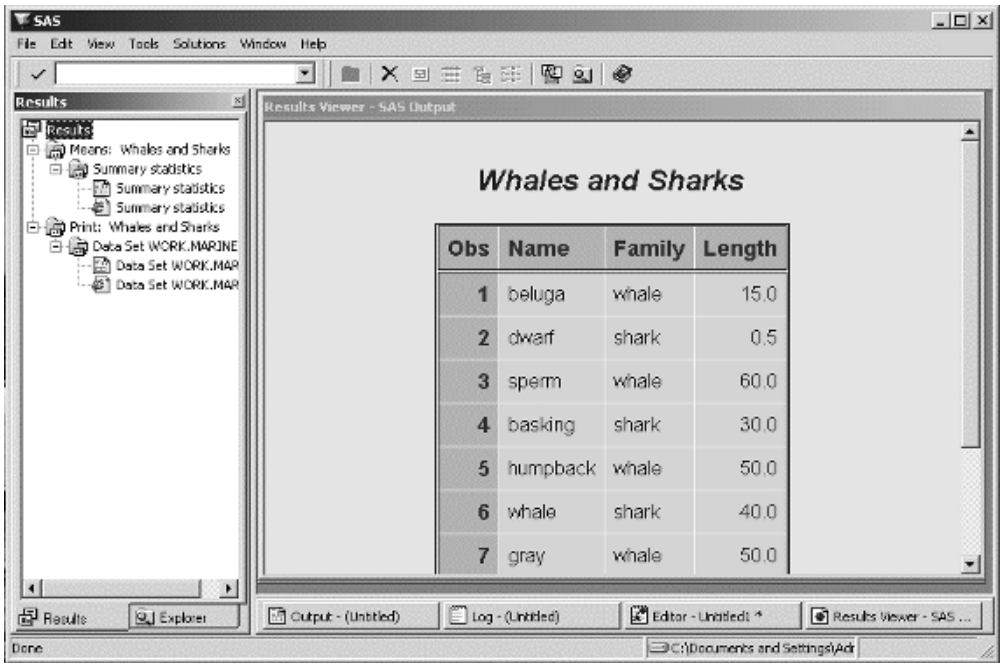
参数选择窗口 选择菜单栏中工具（tools）下拉菜单的选项（options）-参数选择（preferences）。选择结果（result）选项卡，如下图所示：



这个参数选择窗口上，有一个“创建列表（create listing）”选项，默认的输出就是列表输出。下面还有“创建 HTML”选项，用来创建 HTML。样式（style）选项用来为 HTML 选择一个风格样式。选择之后，点击 OK。

结果浏览窗口和结果窗口 （注意下面提到的三种窗口：结果浏览窗口 result viewer，结果窗口 results window，输出窗口 output window）

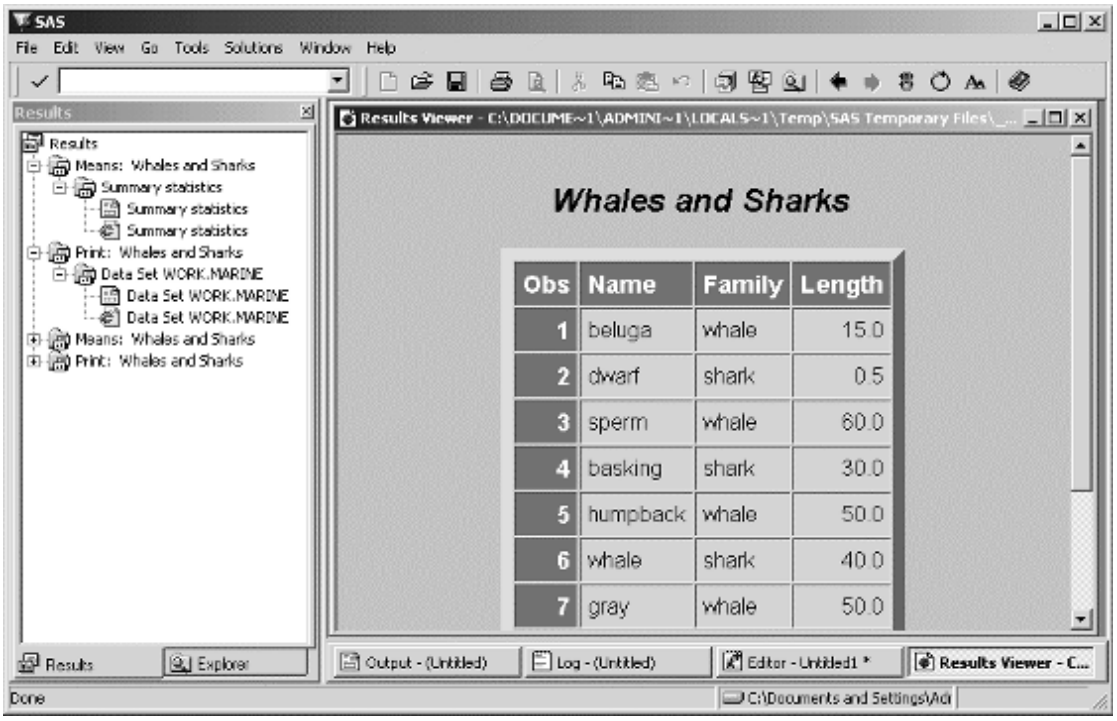
一旦选择了 HTML 输出，每次运行程序的时候都会自动出现一个结果浏览窗口（result viewer），下图显示了在运行了一个均值和输出的程序后，出现两个窗口：结果浏览窗口——显示 HTML 输出，结果窗口——显示输出的目录。



结果浏览窗口一次只显示输出结果的一个部分，通过选择结果窗口的目录，可以查看其他的部分。值得注意的是，结果窗口给出的目录中，每一个部分都给出了两种相同的目录，一个是链接到输出窗口，一个是链接到结果浏览窗口

保存结果浏览窗口的输出（output of result viewer）的方法：激活结果浏览窗口，选择菜单栏文件（file）下拉菜单的保存（save as）、输出（print）。

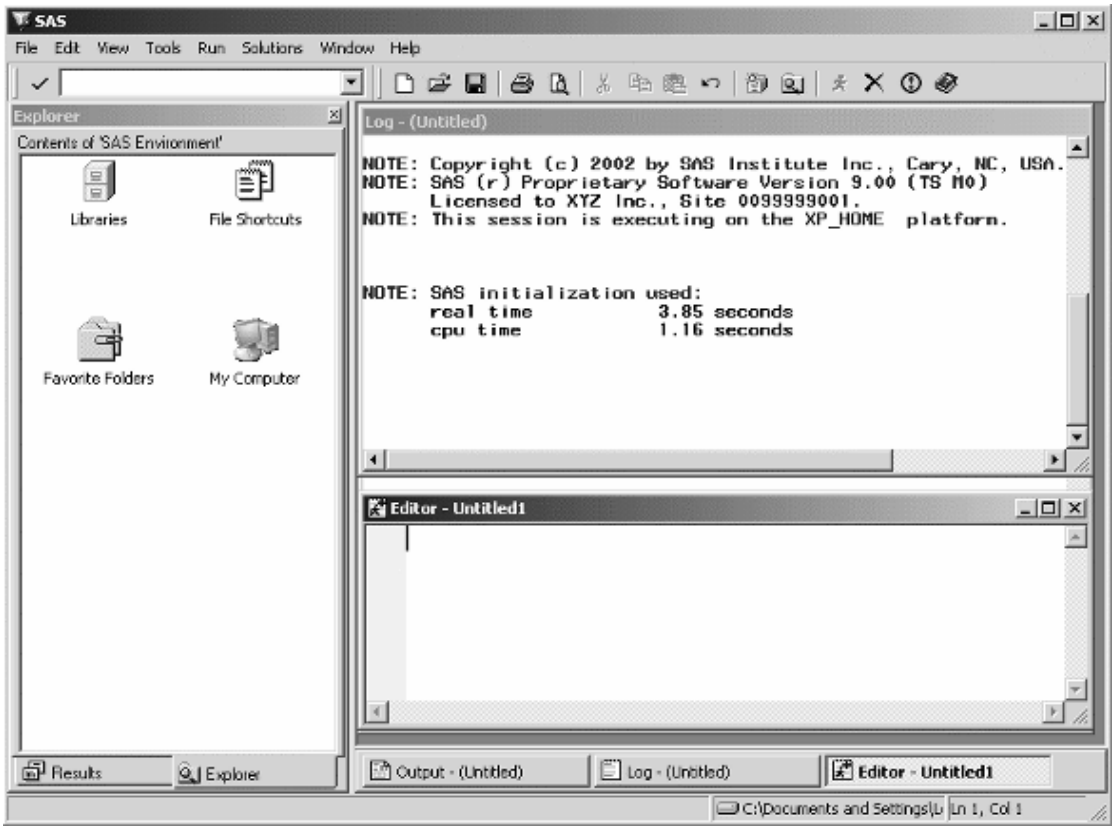
可以在 style 中为输出选择不同的风格，如下图就是 D3D 的风格：



1.11 SAS 数据的逻辑库

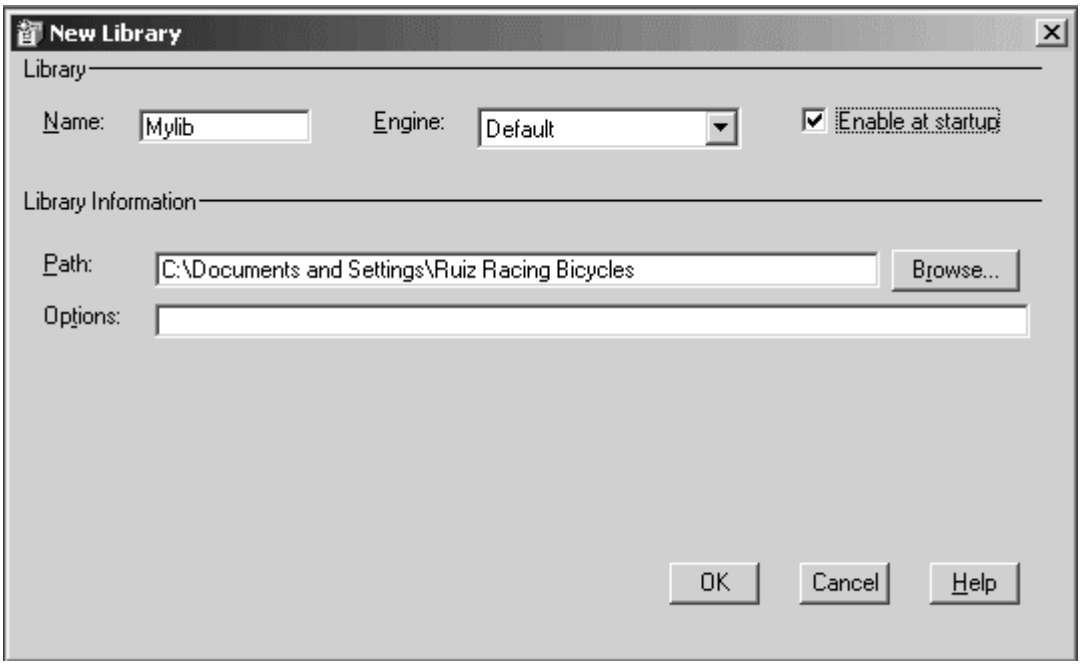
SAS 逻辑库库只是数据集文件存放的地点。打开 SAS 视窗模式后，会看到 SAS 资源管理器窗口（explore），双击逻辑库图标，资源管理器窗口会显示出所有已定义的图标。要返回前一级窗口，选择查看（view）-向上一级（up one level），或者在工具栏中直接点击向上一级图标 

逻辑库窗口 打开逻辑库窗口后，除了自己创建的库外，至少会呈现三个逻辑库：sashelp，sasuser，和 work，如果安装了某些 SAS 模块，还有一些特别的逻辑库，比如 SAS/GRAPH 模块的 Maps 逻辑库。Sashelp 包含了控制 SAS 会话以及样本数据集的一些信息。WORK 是 sas 数据集的临时储存地点，创建的数据集如果没有指定库，则默认储存在这里，关闭程序时则自动删除数据集。也可以更改默认的库，从而不是临时库。



创建新逻辑库 创建新逻辑库有两种方法：在逻辑库窗口中选择文件（file）下拉菜单的新建（new）；或者直接右键——新建。

在新建逻辑库（new library）窗口中，为你的逻辑库起一个名字，这里叫做 Mylib,指定库的存放路径。如果不想每次启动 SAS 都要调用这个库，则勾选启动时启用（enable at startup）即可。

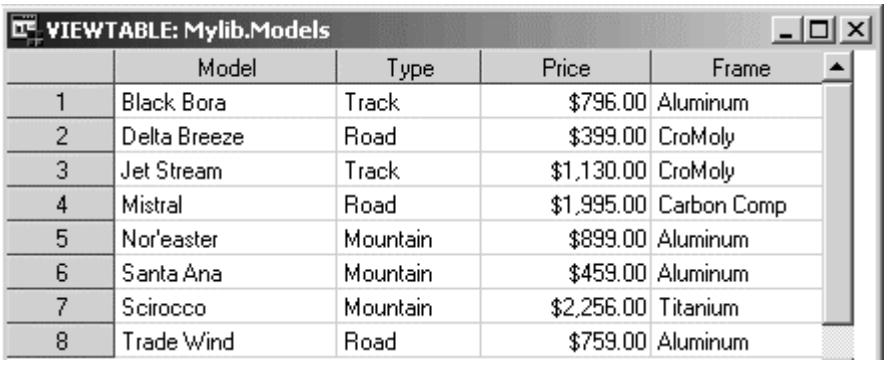


这是一个新建的 Mylib 逻辑库的视图。

1.12 用 SAS 资源管理器访问 SAS 数据集

可以利用 SAS 资源管理器打开数据集、浏览、编辑，也可以列出数据集的信息，如创建时间和变量名。

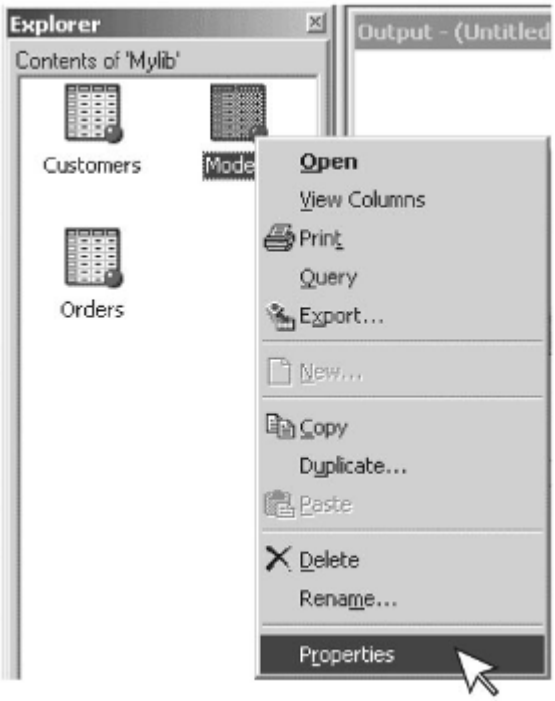
目录窗口 打开一个逻辑库，进入一个目录窗口，显示这个库中所有的文件和文件夹。右图的目录窗口中显示 Mylib 逻辑库有三个文件：Customers，Models，orders。双击某文件，则可以打开这个文件的可视视图。

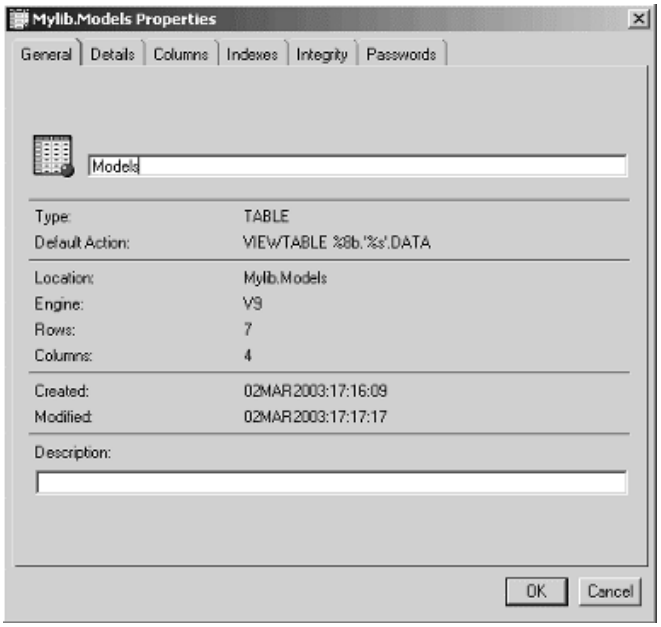


	Model	Type	Price	Frame
1	Black Bora	Track	$796.00	Aluminum
2	Delta Breeze	Road	$399.00	CroMoly
3	Jet Stream	Track	$1,130.00	CroMoly
4	Mistral	Road	$1,995.00	Carbon Comp
5	Nor'easter	Mountain	$899.00	Aluminum
6	Santa Ana	Mountain	$459.00	Aluminum
7	Scirocco	Mountain	$2,256.00	Titanium
8	Trade Wind	Road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

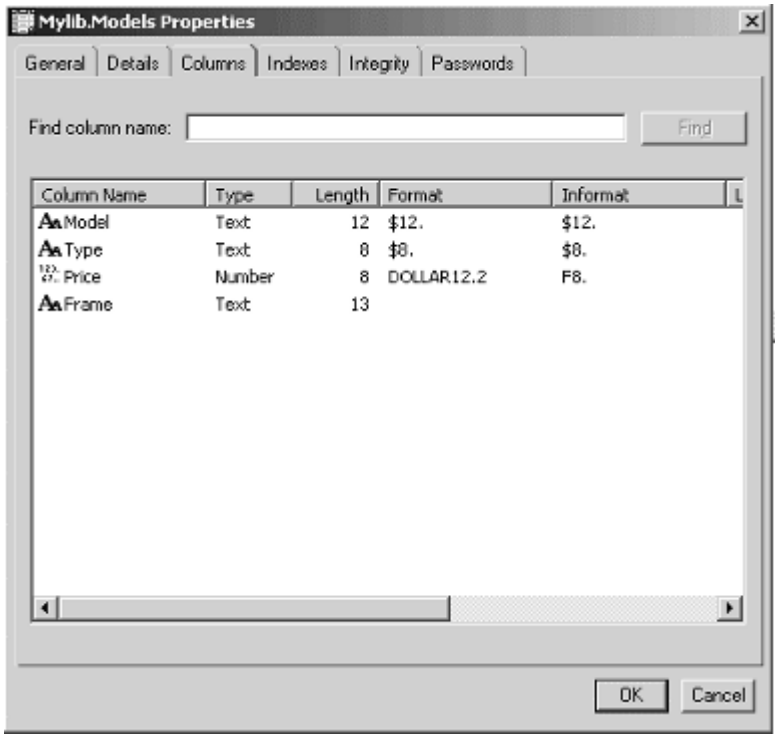
可视视图 这个窗口允许你创建、浏览、编辑数据集。

列出 SAS 数据集的属性 用资源管理器还可以列出 SAS 数据集的一些属性情况，右击某文件，选择下拉菜单的属性（properties）





属性窗口显示了 SAS 数据集的属性信息，如创建时间、行列数等。



如果选择列（columns）选项卡，则出现数据的列信息

1.13 使用 SAS 系统选项

SAS 系统选项是影响 SAS 运行的一些参数，比如输出的显示、内存的占用、错误的处理等问题。这些小问题由 SAS 为你设定好，你也可以改变它。

SAS 系统选项的参数不是所有都适合你的操作环境，适合于你的再 SAS 的帮助文档中给出。可以通过打开 SAS 系统选项窗口或使用 option 程序来查看你的 SAS 系统参数。Option 程序的语句如下：

```
Proc options;
Run;
```

有四种方法可以指定系统选项的参数，SAS 帮助文档会告诉你哪种适合你的操作系统：

1. 系统管理员会创建一个包含了系统选项设定的配置文件，每次 SAS 启动时都会访问这个文件。

2. 在启动 SAS 之后，根据系统提示指定系统选项。

3. 如果使用 SAS 视窗环境，可以从 SAS 系统选项中改变已选择的选项。

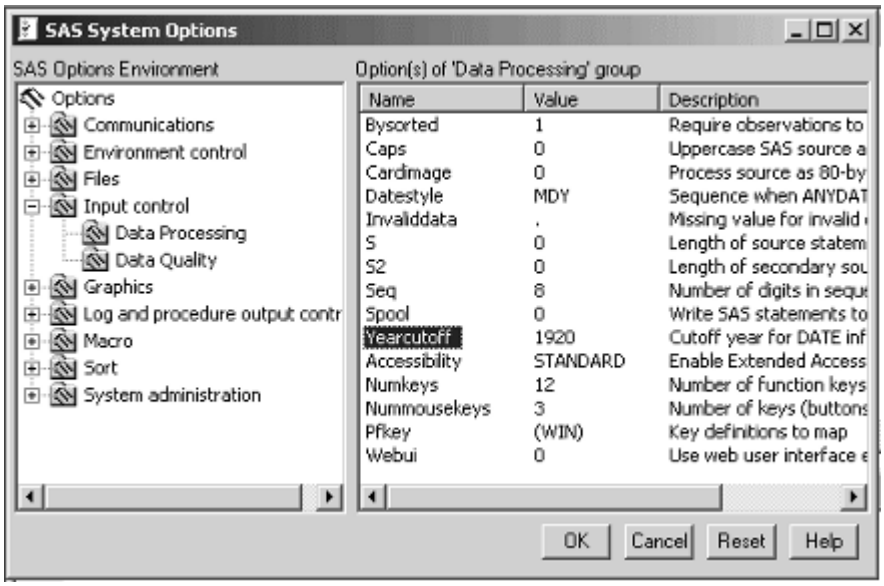
4. 使用 OPTIONS 语句。

这四种方法按照优先性排列，方法 2 优先于方法 1，方法 3 优先于方法 2，方法 4 优先于 3。如果使用 SAS 视窗环境，方法 3,4 最好。

OPTIONS 语句 OPTIONS 语句是 SAS 程序的一部分，并可影响之后的所有语句。由 OPTIONS 关键词开头，后面是相关选项。比如：

OPTIONS LINESIZE=80 NODATE;

OPTIONS 语句既不属于数据步也不属于过程步，这个全局变量可以出现在程序的任何部分，但放在开头最有意义，你可以很容易看到哪些选项在发挥作用。如果 OPTIONS 语句只出现在数据步或者过程步中，那么它会影响那个过程，和下面的过程（If the OPTIONS statement is in a DATA or PROC step,then it affects that stepand the following steps.）。注意，后面的 OPTIONS 语句会覆盖前面的，即以后面的 OPTIONS 为主。



SAS 系统选项窗口 通过这个窗口也可以改变系统选项。可以通过在命令栏中输入“OPTIONS”，或从工具（tools）下拉菜单中选择.

窗口出来后，找到要修改的部分，右击——修改值（modify the value）即可。

一般选项 下面是一些可能用到的一般系统选项

CENTER|NOCENTER	输出是否居中，还是左对齐。默认居中
DATE|NODATE	今天的日期是否出现在输出的顶部。默认输出
LINESEZE=n	控制输出行的最大长度，n 可能的值为 64 到 256
NUMBER|NONUMBER	输出的页面页码是否需要。默认需要。
ORIENTATION=PORTRAIT； ORIENTATION=LANDSCAPE；	指定打印输出的方向。默认竖向（portrait）
PAGENO=n	输出页开始的页面。默认为 1
PAGESIZE=n	每个页面输出的最大行数。可能的值为 15 到 32767

RIGHTMARGIN=n LEFTMARGIN=n TOPMARGIN=n BOTTOMMARGIN=n	指定打印输出的边缘大小。默认 0.00 英寸（Specifies size of margin(such as 0.75in or 2cm)to be used for printing output.Default:0.00in.）
YEARCUTOFF=yyyy	设定起始年份

第二章 将你的数据放入 SAS（2.1-2.11）

2.1 将你的数据放入 SAS 的方法

你可能有各种形式的数据，包括手写在纸上、存放在电脑上、或是在数据库管理系统里，不论如何，总有一种方法可以让 SAS 来读取。

SAS 读取的数据的方法主要有以下几种类型：

- 直接输入；
- 从原始数据文件中创建一个 SAS 数据集（creating SAS data sets from raw data files）；
- 将其他软件中的数据文件转换成 SAS 数据集；
- 直接读取其他软件的数据集；

直接输入

- Viewtable 窗口可以让你以表格形式输入数据，可以定义变量、设置属性，如 name、length 和 type(character or numeric).
- SAS 企业向导模块，a Windows only application,has a data entry window that is very similar to the Viewtable window.As with Viewtable,you can define variables and give them attributes.
- SAS/FSP 模块，是 Full Screen Product 的简称，可以设计定制的数据输入窗口，也有检测数据输入错误的功能（The SAS/FSP product is licensed separately from Base SAS software.）。

从原始数据文件中创建一个 SAS 数据集 你有两种方法读取原始数据文件：

- 数据步可以读取任何形式的原始数据文件，这种方法还将在 2.4 中详解。
- 导入向导（Import Wizard）、导入过程（IMPORT procedure）适用于 UNIX、OpenVMS 和 Windows 操作环境的简单方法，可以读取 CSV（comma-separated values）和其他一些限定的文件类型。

将其他软件中的数据文件转换成SAS数据集 如果数据在一个软件中以某种格式存放，但需要用另一种软件分析时，就会很麻烦。有几种方法可以将某种软件中的数据转换成 SAS 数据集：

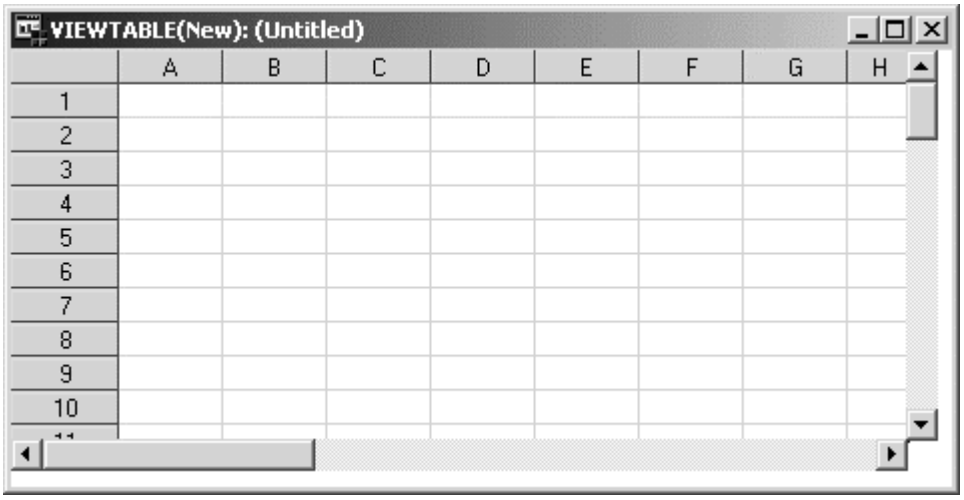
- 如果安装 SAS/ACCESS 模块，可以用导入过程（import procedure）和导入向导（Import Wizard）将 Excel、Lotus、dBase 和 Access 文件导入 SAS 数据集，见 2.3 和 2.17（？）。
- 如果没有安装，可以用存放数据的软件创建一个原始文件，并用数据步或导入过程（import procedure）读取。很多软件都可以创建 CSV 文件。
- Windows 操作环境下也可以用动态数据交换技术（Dynamic Data Exchange，DDE），见 2.18。前提是必须有一个其他的 Windows 程序与 SAS 同时运行，再使用 DDE 和数据步。

直接读取其他软件的数据集

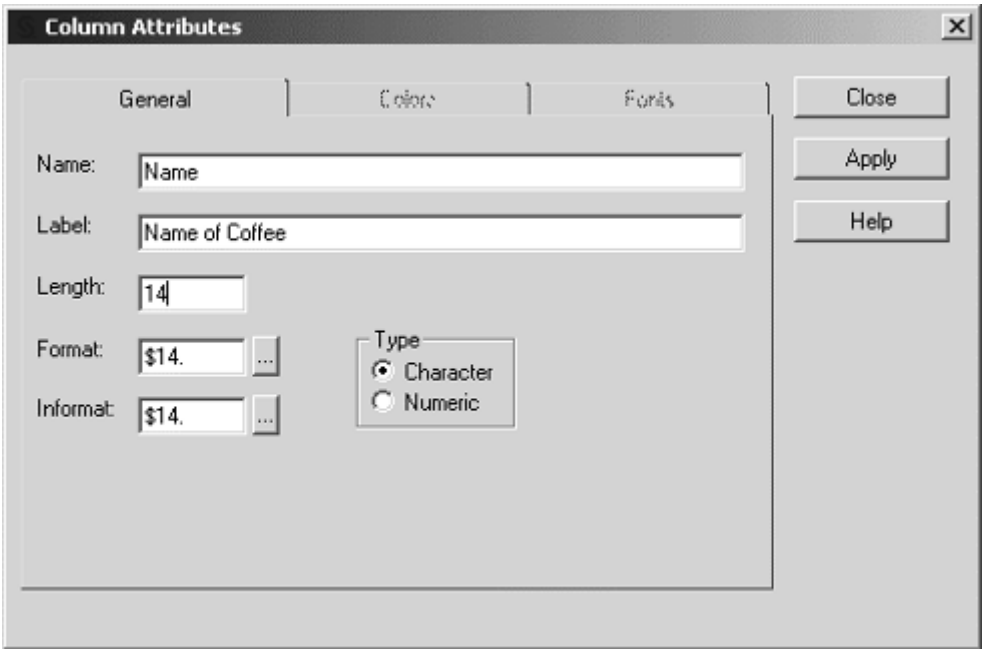
- SAS/ACCESS 产品可以不用转换数据格式读取数据，并适用于大部分大部分数据库管理系统，包括 ORACLE,DB2,INGRES 和 SYBASE（但使用方法本书没有介绍）。
- 使用 Excel engine 和 Access engine 来读取这两种类型的数据。（SAS 帮助文档）
- 还有其他的一些数据引擎（data engines）来读取数据，如 SPSS engine（附录 D），查找帮助文档找到适合你操作环境的所有有效 engine。

2.2 用 Viewtable 窗口输入数据

调用 Viewtable 窗口，在工具栏的下拉菜单中选择表编辑器（Table Editor）。



列属性窗口 每一列顶部的字母是默认的变量名，右击变量名，即可打开列属性（column attributes）窗口，设置属性之后，点击应用（apply），设置完后关闭即可。



输入数据

保存表 选择文件（file）——另存为（save as），选择一个逻辑库保存，如果想保存在新的逻辑库，点击创建新逻辑库图表（New Library），输入逻辑库的名字和保存路径。

打开一个已有的表 首先打开表编辑器，选择文件（file）——打开（open）。打开之后，SAS 默认的是浏览模式，如果要对数据进行编辑，则要在编辑（edit）菜单中选择编辑模式。也可以在资源管理器窗口中双击打开一个已有的表。

其他功能 其他一些功能包括排序、打印（printing）、增加和删除行、一次浏览一行（窗体视图 Form View）、一次浏览多行（表视图 Table View）。图标和菜单都可以选择这些功能。

在 SAS 程序中使用表 如下程序语句可以将表内容输出打印：

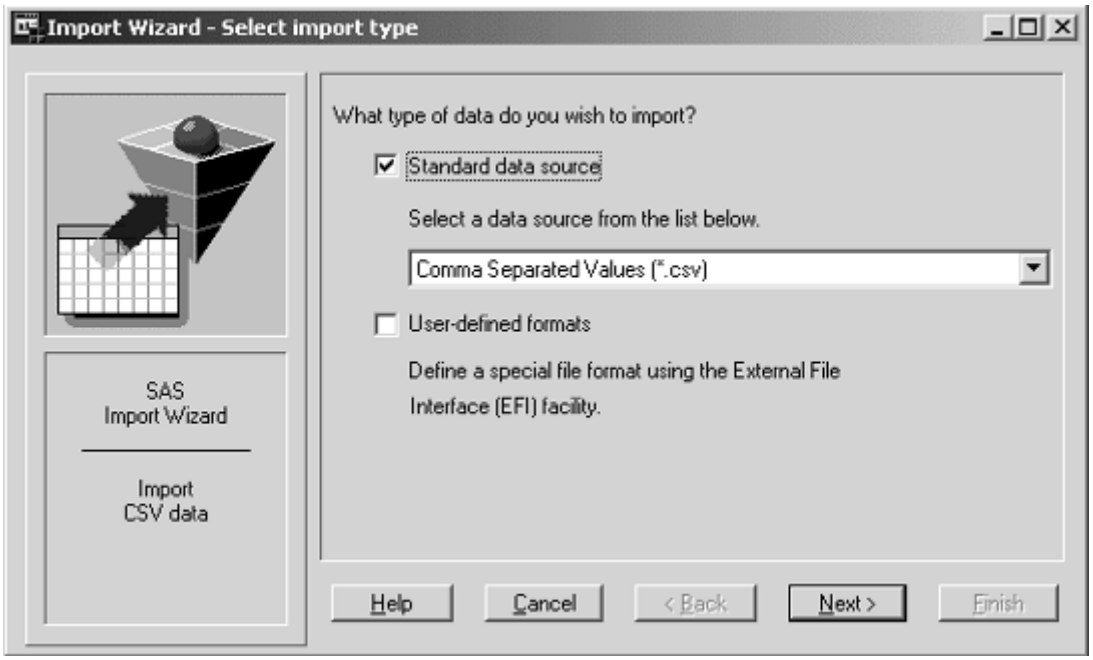
```
PROC PRINT DATA=Sasuser.coffee;
```

RUN；

2.3 用导入向导（Import Wizard）读取文件

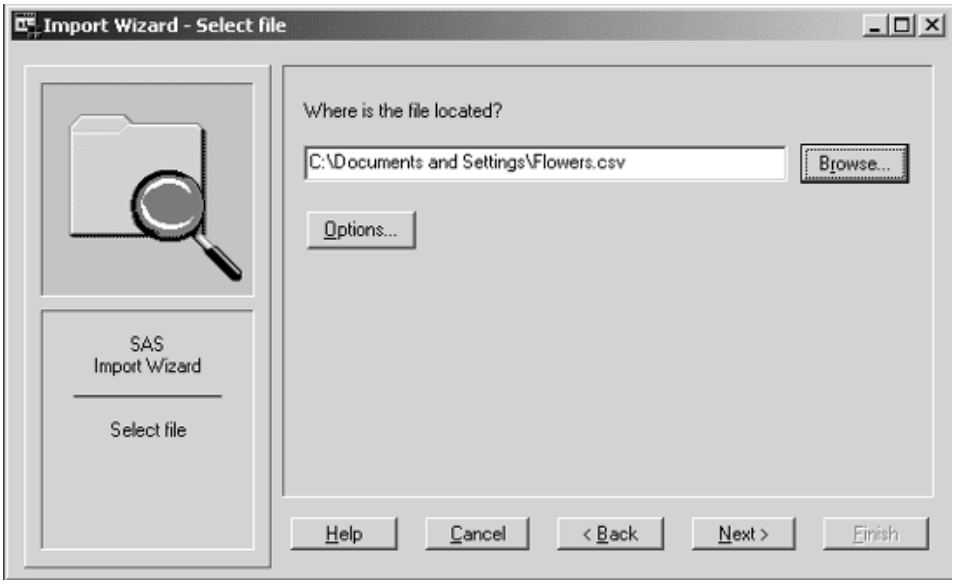
导入向导会浏览你的文件以决定变量的类型，并默认数据表的第一行存放变量名。

第一步，从文件（file）下拉菜单中选择导入数据（import data）导入向导（Import Wizard）

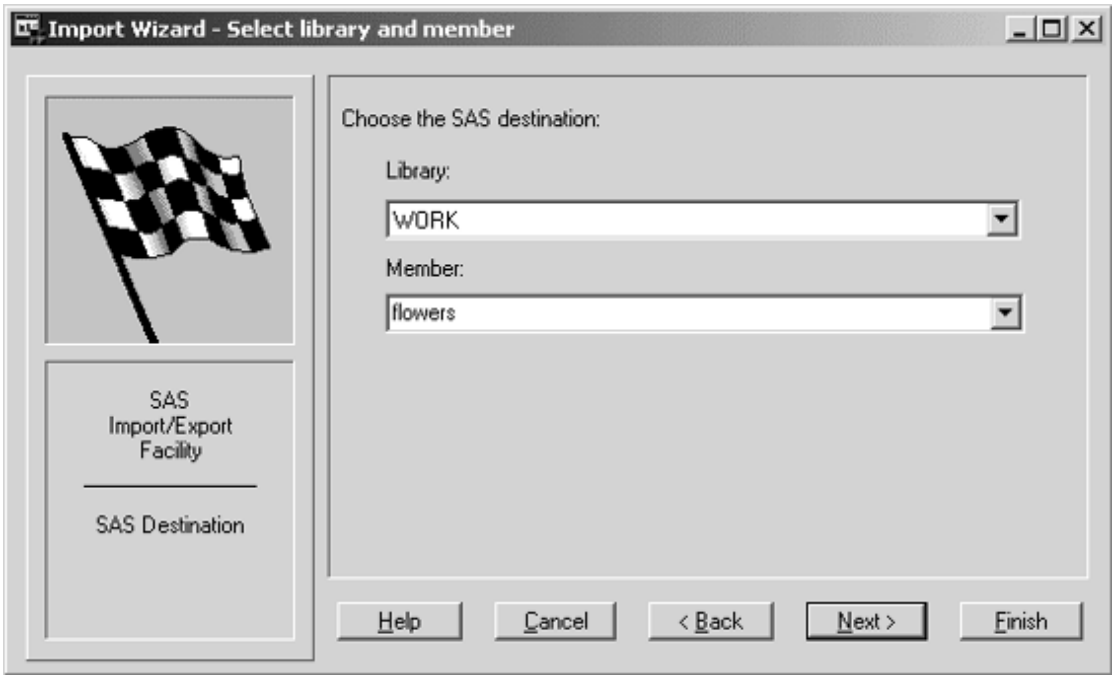


第二步，从 standard data sources 中选择要导入的数据类型。以 comma separated values（*.csv）为例。点击下一步

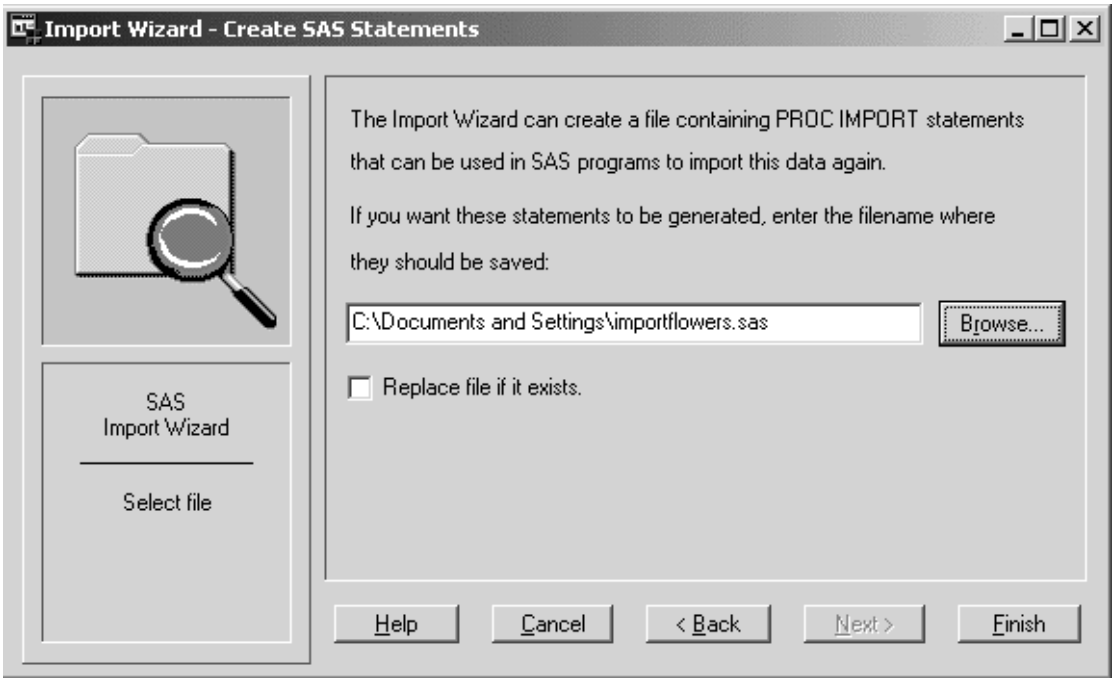
第三步，指定要导入的文件位置，SAS 默认第一行存放变量名，从第二行开始存放数据。Options 可以改变这种默认选择。



第四步，选择数据集要存放的逻辑库，并为数据集取一个名字（member）



最后，导入向导创造一个 proc import 语句，可以是 SAS 再次导入这个数据。



另外，对于一些类型的数据文件还有额外的步骤，比如 Microsoft Access 文件，你需要输入数据库名和药导入的表名，有时甚至还要输入 user 的 ID 号和密码。

在 SAS 程序中使用导入数据 比如你将数据存放在 work 逻辑库，并署名为 flowers，那么你可以这样来输入它：

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

由于 work 是默认的逻辑库，所以也可以直接这样：

```
PROC PRINT DATA=flowers;
RUN;
```

2.4 告诉 SAS 你的原始数据在哪

如果数据是原始数据（比如 text,ASCII,sequential,flat files），那么用数据步来读取能带给你最大的灵活性。但首先你要告诉 SAS 你的原始数据在哪。

原始数据可以通过文本编辑器（text editors）或系统命令。对 PC 使用者来说，原始数据没有相关联的应用程序（就像 doc 文件与 word 相关联，双击 doc 程序就默认调用 word 程序以打开），有时他们会与像 Microsoft Notepad 这样的简单编辑器相关联。

内部原始数据 如果直接将数据输入 SAS 程序中，那么数据就是 SAS 内部数据。DATALINES 是一个指示，告诉 SAS 下面跟着是数据行，直到分号结尾，这个分号可以另起一行，也可以接在数据的后面。也可以用 card 代替 DATALINES。下面是一个程序，这个程序创建了一个 USPRESIDENTS 数据集。（Input 语句还将在 2.5 和 2.15 中讨论）

```
* Read internal data into SAS data set 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 数据外 SAS 程序外部时，使用 INFILE 语句告诉 SAS 外部数据的文件名和存放路径，它在 data 语句之后，在 INPUT 语句之前。INFILE 后面的文件名和路径要用引号，各种系统的引用方式各不同：

Windows:	`INFILE 'c:\MyDir\President.dat';`
UNIX:	`INFILE '/home/mydir/president.dat';`
OpenVMS:	`INFILE '[username.mydir]president.dat';`
OS/390 or z/OS:	`INFILE 'MYID.PRESIDEN.DAT';`

假设有一个 President.dat 数据在你的 C 盘的 MyRawData 目录中，

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

那么可以用如下方式引用这个外部数据：

```
* Read data from external file into SAS data set;
DATA uspresidents;
   INFILE 'c:\MyRawData\President.dat';
   INPUT President $ Party $ Number;
RUN;
```

SAS 日志 读取外部数据时，SAS 日志会给一些很有用的信息，检查这些信息可以解决一些问题，比如对比 INFILE 语句读取的记录数和数据集中实际观测值，可以确定 SAS 是否正确的读取了数据。上面程序的日志如下图所示，

```
NOTE: The infile 'c:\MyRawData\President.dat' is:
      File Name=c:\MyRawData\President.dat,
      RECFM=V,LRECL=256
NOTE: 4 records were read from the infile 'c:\MyRawData\President.dat'.
      The minimum record length was 13.
      The maximum record length was 13.
NOTE: The data set WORK.USPRESIDENTS has 4 observations and 3 variables.
```

过长记录 在一些操作环境中，SAS 假定外部数据文件的记录长度为 256 或更少（记录长度是指某行中的字符数，包括空格），如果长度过长，SAS 不能读取全部，此时需要在 INFILE 语句中使用 LRECL=来指定长度，这个长度必须是数据中最长行的长度，如：

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

可以通过 SAS 日志来查看最大记录长度。

2.5 list input 读取空格分开的原始数据

如果原始数据都至少被一个空格分开，那么用 list input 读取数据可能是正确的。List input 是一个很简单的读取数据的方式，但是会受到很多限制。你必须读取所有的数据记录，不能跳过某些值、缺失值必须用句号“.”代替。字符串数据不能包含空格、长度不能超过 8 个字符。而且如果数据文件包含日期变量或者其他需要特别处理的变量，list input 将不再适用。虽然很多限制，但仍有大量的文件可以用这种方式读取。

INPUT 语句是数据步的一部分，它告诉 SAS 如何读取原始数据。使用 List input 来写 input 语句：在 INPUT 关键词之后列出变量名（按照变量在文件中出现的顺序），变量名长度在 32 个字节（含）以下，只能包含字母、下划线和数据，并必须以字母或者下划线开头。如果变量是字符串，后面要价“$”号，值与值之间至少有一个空格，语句要以分号结束。如：

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

这表明输入三个变量，其中 name 是字符串，age 和 height 是数值变量。

例子 你想组织一次青蛙跳跃比赛，现在你记录了每只参赛青蛙的名字、体重、和三次跳跃的距离，如果某次的跳跃距离不合格，那么就用“.”代替，数据文件 ToadJump.dat 形式如下：

```
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 . . .
```

虽然不是很整洁、但满足 list input 的所有要求（字符串长度小于 8 个字节、不包含空格、值之间都有至少一个空格，缺失数据也用句号代替）。Nosiy 的数据溢出到第二行了，但这不影响，SAS 会按照变量顺序自动跳到下一行读取。如下是读取这个数据的 SAS 程序：

```
* Create a SAS data set named toads;
* Read the data file ToadJump.dat using list input;
DATA toads;
   INFILE 'c:\MyRawData\ToadJump.dat';
   INPUT ToadName $ Weight Jump1 Jump2 Jump3;
* Print the data to make sure the file was read correctly;
PROC PRINT DATA = toads;
   TITLE 'SAS Data Set Toads';
RUN;
```

Input 后面是变量名，ToadName 是字符串变量，其他是数值变量；proc print 过程用来输出数据集中所有的变量和观测值；title 语句用告诉 SAS 输出顶部的标题，如果不指定标题，SAS 将以“the SAS system”作为标题在每一页的顶部。输出的形式如下：

SAS Data Set Toads 1

Obs	Toad Name	Weight	Jump1	Jump2	Jump3
1	Lucky	2.3	1.9	.	3.0
2	Spot	4.6	2.5	3.1	0.5
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 Noisy 的数据溢出到下一行，因此下面的说明会出现在 SAS 日志上：

```
NOTE: SAS went to a new line when INPUT statement reached past the end of a line.
```

2.6 column input 读取按固定列排列的原始数据

当一些原始数据的值之间没有空格分开，或者没用用句号代替缺失值时，list input 就不能用。

但当每个变量的值都出现在数据行的相同位置时，并且变量值是字符串或者标准数值（只包含数据、小数点、正负号、和科学标注的 E。逗号和日期都不能算）时，可以使用 column input 来读取。

相比 list input，column input 有如下优势：

- 不要求变量值之间的空格；
- 缺失值可以直接用空格代替；
- 字符串中可以包含空格；
- 可以跳过不需要的变量。

调查数据使用 column input，因为调查答案的记录都是用单个数字（0-9），如果每个答案之间再用空格分开，就会使整个文件会扩大两倍。有地址的数据文件也使用 column input，因为地址之中常常包含空格，比如街道 Martin Luther King Jr.Boulevard 在 column input 中就可以当成一个变量而不是五个。可能用 column input 读取的数据也可以用 formatted input 读取或者几种方式组合。

Column input 的 input 语句格式如下：input 关键字后接变量名、再接变量的列位置（列位置是字符或者数值在一行中的位置）。字符串变量名后仍要用“空格+$”，变量名之间仍要用空格隔开。示例如下：

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

这个语句表明，Name 变量，在行中占据第 1 列第 10 列，为字符串变量，age 占据第 11-13 列，为数值变量，height 占据第 14-18 列，数值变量。

例子 原始数据记录如下：

```
----+----1----+----2----+----3----+----4
Columbia Peaches      35  67  1 10  2  1
Plains Peanuts       210      2  5  0  2
Gilroy Garlics        151035 12 11  7  6
Sacramento Tomatoes  124  85 15  4  9  1
```

读取这个数据的 column input 程序如下：

```
* Create a SAS data set named sales;
* Read the data file Onions.dat using column input;
DATA sales;
   INFILE 'c:\MyRawData\Onions.dat';
   INPUT VisitingTeam $ 1-20 ConcessionSales 21-24 BleacherSales 25-28
         OurHits 29-31 TheirHits 32-34 OurRuns 35-37 TheirRuns 38-40;
* Print the data to make sure the file was read correctly;
PROC PRINT DATA = sales;
   TITLE 'SAS Data Set Sales';
RUN;
```

第一个变量 visitingteam 占据第 1-20 列，为字符变量；concessionsales 占据第 21-24 列，为数值变量，下面几个变量均占据固定的列。输出结果如下：

SAS Data Set Sales 1

Obs	VisitingTeam	Concession Sales	Bleacher 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

2.7 informats 读取非标准格式的原始数据

有时候原始数据不全是字符串或者数值，比如类似 1,00,001 这样包括逗号的字符串值，电脑就不能读取，其他诸如包含美元符号、十六进制、压缩十进制的数据都是非标准数据。SAS 中，informats 可以用来告诉电脑如何读取这样的数值。

日期是最普通的非标准数据，SAS informats 会把类似 10-31-2003 或者 31OCT03 转换成数字，日期的起点为 1960 年 1 月 1 日，即这一天的数字为 0。

Informats 的三种普遍格式为：字符串、数值、日期。这三种格式的形式如下：

Character	Numeric	Date
$informatw.	informatw.d	informatw.

$代表是字符串、informats 代表形式（比如日期的 MMDDYY）、w 是宽度、d 是小数点的位数、最后是句号“.”，缺少句号会使得 SAS 把形式（如 MMDDYY）当做变量名。一个简单 formatted input 的简单 INPUT 语句如下：

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

Name 为字符串变量，占据 10 个宽度，即列位置从 1-10；age 为数值变量，占据 3 个宽度，列位置从 11 到 13；height 也为数值变量，占据 5 个宽度，包括了 1 位小数点和小数点本身，列位置从 14-18（如 150.3）；最后是日期变量，从第 19 列开始，形式为 MMDDYY。

例子 原始数据如下：

```
Alicia Grossman  13 c 10-28-2003 7.8 6.5 7.2 8.0 7.9
Matthew Lee       9 D 10-30-2003 6.5 5.9 6.8 6.0 8.1
Elizabeth Garcia 10 C 10-29-2003 8.9 7.9 8.5 9.0 8.8
Lori Newcombe     6 D 10-30-2003 6.7 5.6 4.9 5.2 6.1
Jose Martinez     7 d 10-31-2003 8.9 9.510.0 9.7 9.0
Brian Williams   11 C 10-29-2003 7.8 8.4 8.5 7.9 8.0
```

读取这个数据的 informat 程序如下：

```
* Create a SAS data set named contest;
* Read the file Pumpkin.dat using formatted input;
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);
* Print the data set to make sure the file was read correctly;
PROC PRINT DATA = contest;
   TITLE 'Pumpkin Carving Contest';
RUN;
```

年龄后面的+1 代表跳过一列，即原始数据中年龄后面有一个空格。最后的 5 个变量 score1-score5，都要求有同样的形式，4.1。将变量名和形式分别放在两个括号集中，可以一次性定义很多变量。输出结果如下：

Pumpkin Carving Contest 1

Obs	Name	Age	Type	Date²	Score1	Score2	Score3	Score4	Score5
1	Alicia Grossman	13	c	16006	7.8	6.5	7.2	8.0	7.9
2	Matthew Lee	9	D	16008	6.5	5.9	6.8	6.0	8.1
3	Elizabeth Garcia	10	C	16007	8.9	7.9	8.5	9.0	8.8
4	Lori Newcombe	6	D	16008	6.7	5.6	4.9	5.2	6.1
5	Jose Martinez	7	d	16009	8.9	9.5	10.0	9.7	9.0
6	Brian Williams	11	C	16007	7.8	8.4	8.5	7.9	8.0

2.8 可选择变量形式

一般使用的变量形式的定义，以及它们的宽度范围和默认宽度如下：

Informat	Input data	INPUT statement	Results
Character			

$CHAR*w*.	my cat
	  my cat
		INPUT Animal $CHAR10.;	my cat
			  my cat

$HEX*w*.	6C6C	INPUT Name $HEX4.;	11 (ASCII) or %% (EBCDIC)
$*w*.	my cat my cat	INPUT Animal $10.;	my cat my cat
Date, Time, and Datetime			
DATE*w*.	1jan1961 1 jan 61	INPUT Day DATE10.;	366 366
DATETIME*w*.	1jan1960 10:30:15 1jan1961,10:30:15	INPUT Dt DATETIME18.;	37815 31660215
DDMMYY*w*.	01.01.61 02/01/61	INPUT Day DDMMYY8.;	366 367
JULIAN*w*.	61001 1961001	INPUT Day JULIAN7.;	366 366
MMDDYY*w*.	01-01-61 01/01/61	INPUT Day MMDDYY8.;	366 366
TIME*w*.	10:30 10:30:15	INPUT Time TIME8.;	37800 37815
Numeric			
COMMA*w.d*	$1,000,001 (1,234)	INPUT Income COMMA10.;	1000001 -1234
HEX*w*.	F0F3	INPUT Value HEX4.;	61683
IB*w.d*	[redacted]	INPUT Value IB4.;	255
PD*w.d*	[redacted]	INPUT Value PD4.;	255
PERCENT*w*.	5% (20%)	INPUT Value PERCENT5.;	0.05 -0.2
w.d	1234 -12.3	INPUT Value 5.1;	123.4 -12.3

Informat	Definition	Width range	Default width
Character			
$CHAR*w*.	Reads character data—does not trim leading or trailing blanks	1-32,767	8 or length of variable
$HEX*w*.	Converts hexadecimal data to character data	1-32,767	2
$*w*.	Reads character data—trims leading blanks	1-32,767	none
Date, Time, and Datetime*			
DATE*w*.	Reads dates in form: *ddmmmyy* or *ddmmmyyyy*	7-32	7
DATETIME*w*.	Reads datetime values in the form: *ddmmmyy hh:mm:ss.ss*	13-40	18
DDMMYY*w*.	Reads dates in form: *ddmmyy* or *ddmmyyyy*	6-32	6
JULIAN*w*.	Reads Julian dates in form: *yyddd* or *yyyyddd*	5-32	5
MMDDYY*w*.	Reads dates in form: *mmddyy* or *mmddyyyy*	6-32	6
TIME*w*.	Reads time in form: *hh:mm:ss.ss* (hours:minutes:seconds—24-hour clock)	5-32	8
Numeric			
COMMA*w.d*	Removes embedded commas and $, converts left parentheses to minus sign	1-32	1
HEX*w*.	Converts hexadecimal to floating-point values if *w* is 16. Otherwise, converts to fixed-point.	1-16	8
IB*w.d*	Reads integer binary data	1-8	4
PD*w.d*	Reads packed decimal data	1-16	1
PERCENT*w*.	Converts percentages to numbers	1-32	6
w.d	Reads standard numeric data	1-32	none

2.9 混合读取方式

每种数据读取方式都有其优势，list 最简单，column 和 formatted 虽然复杂但是不要求变量之间的空格，并且变量名中可以包含变量，而且 formatted 可以读取特殊的数据比如日期。SAS 可以灵活的让你搭配不同的读取方式，以达到最大的方便。

例子 如下的原始数据记录了美国国家公园的信息：姓名（name）、所属周（state）、建立时间（year established）、占地面积（size inacre）：

```
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
```

有多种方式进行数据读取，下面的程序是方式之一：

```
* Create a SAS data set named nationalparks;
* Read a data file Park.dat mixing input styles;
DATA nationalparks;
   INFILE 'c:\MyRawData\Park.dat';
   INPUT ParkName $ 1-22 State $ Year @40 Acreage COMMA9.;
PROC PRINT DATA = nationalparks;
   TITLE 'Selected National Parks';
RUN;
```

其中 ParkName 是 column 方式读取，State 和 Year 是 list 方式读取，Acreage 是 formatted 方式读取，输出结果如下所示：

Selected National Parks 1

Obs	ParkName	State	Year	Acreage
1	Yellowstone	ID/MT/WY	1872	4065493
2	Everglades	FL	1934	1398800
3	Yosemite	CA	1864	760917
4	Great Smoky Mountains	NC/TN	1926	520269
5	Wolf Trap Farm	VA	1966	130

混合读取方式有时会遇到问题：SAS 通过一个指示器标注位置，来读取原始数据的一行，但每种读取方式对指示器的使用稍有不同。List 方式下，SAS 自动找到非空格区域并开始读取；column 方式下，SAS 读取你所指定的特定位置；informatted 方式下，SAS 不理会指示器的标准，只是依次的读取。这时，就会需要列指示器@n，来人为的让 SAS 的读取直接跳至某列。

在上面的程序中，列指示器@40 告诉 SAS 在读取 Acerage 变量之前，移动到第 40 列去，如果移去指示器，程序为：

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

输出结果如下图所示：

Selected National Parks 1

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	.

之所以出现这样的结果，要看原始文件的列坐标排列：

```
----+----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
```

Comma9 告诉 SAS 读取 9 列，SAS 就会读取包括空格在内的 9 列，这便会导致输出结果的问题。

2.10 读取凌乱的原始数据

有的数据排列混乱，长度不一。这样的数据需要新的工具处理：@'character' 列指示器和 colon modifier。

@'character' 列指示器 2.9 中提到@column 列指示器可以让 SAS 直接从某列开始读取数据。但有时候你不知道要读取的数据是从哪列开始，此时你只要知道要读取的数据的前面那个字符或单词即可。比如有一个关于狗的原始文件，你想要读取狗的品种号，但文件排列很凌乱，只知道品种号跟随在单词 breed 后面，那么可以用如下方式读取：

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

colon modifier 由于 input 读取字符串变量默认为 8 个字符，因此在上例中如果狗的品种名（dogbreed）超过 8 个字符，则需要定义长度，定义为$length，在该长度中，空格也算在内。如果要使 SAS 读取过程中遇到空格则不再继续读取，则要在$length 前面加冒号":"。比如原始数据中有这么一行：

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

如果用上述不同方法读取，会有不同结果：

Statements	Value of variable DogBreed
INPUT @'Breed:' DogBreed $;	Rottweil
INPUT @'Breed:' DogBreed $20.;	Rottweiler Vet Bill
INPUT @'Breed:' DogBreed :$20.;	Rottweiler

例子 web 日志是凌乱数据的一个很好例子，下面是一个网站的 web 日志，数据开始于访问 IP，后面有访问日期、访问文件名等信息。

```
130.192.70.235 - - [08/Jun/2001:23:51:32 -0700] "GET /rover.jpg HTTP/1.1" 200 66820
128.32.236.8 - - [08/Jun/2001:23:51:40 -0700] "GET /grooming.html HTTP/1.0" 200 8471
128.32.236.8 - - [08/Jun/2001:23:51:40 -0700] "GET /Icons/brush.gif HTTP/1.0" 200 89
128.32.236.8 - - [08/Jun/2001:23:51:40 -0700] "GET /H_poodle.gif HTTP/1.0" 200 1852
118.171.121.37 - - [08/Jun/2001:23:56:46 -0700] "GET /bath.gif HTTP/1.0" 200 14079
128.123.121.37 - - [09/Jun/2001:00:57:49 -0700] "GET /lobo.gif HTTP/1.0" 200 18312
128.123.121.37 - - [09/Jun/2001:00:57:49 -0700] "GET /statemnt.htm HTTP/1.0" 200 238
128.75.226.8 - - [09/Jun/2001:01:59:40 -0700] "GET /Icons/leash.gif HTTP/1.0" 200 98
```

现在想要读取访问日期和访问的文件名，但是它们每行中所占据的列的位置都不同，而且文

件名的长度每行都不一样，那么 SAS 读取这种文件通过如下方式：

```
DATA weblogs;
  INFILE 'c:\MyWebLogs\dogweblogs.txt';
  INPUT @'[' AccessDate DATE11. @'GET' File :$20.;
PROC PRINT DATA = weblogs;
  TITLE 'Dog Care Web Logs';
RUN;
```

@'['作为列指示器，告诉 SAS 读取[之后的内容，@'GET'告诉 SAS 读取 GET 之后的内容，由于文件名作为字符串变量，这里基本都会超过 8 个字节，因此后面附加:$20。输出结果如下：

Dog Care Web Logs 1

Obs	AccessDate[3]	File
1	15134	/rover.jpg
2	15134	/grooming.html
3	15134	/Icons/brush.gif
4	15134	/H_poodle.gif
5	15134	/bath.gif
6	15134	/lobo.gif
7	15135	/statemnt.htm
8	15135	/Icons/leash.gif

2.11 跨行观测值的读取方式

一般原始文件中一行代表一个观测值，有时会出现一个观测值跨行的情况。由于 SAS 会自动转到下一行读取数据，直到读取这个观测的所有变量（input 语句中给出），所以你需要告诉 SAS 什么时候不要换行，以便在日志中不出现 SAS-went-to-a-new-line 的暂停说明，此时需要在 INPUT 语句中加行指示器。（？？？？）

行指示器，斜线/：告诉 SAS 跳至原始数据的第二行；#n：跳至第 n 行，n 代表原始数据中某观测值的行数（#2 则让 SAS 跳至某观测值的第二行），#n 不能用来回跳。

例子 有一组关于温度的数据，temperature.dat 第一行代表城市和州，第二行代表本日最高温和最低温，第三行代表史上最高温和最低温。

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

用如下的程度来读取这份数据：

```
* Create a SAS data set named highlow;
* Read the data file using line pointers;
DATA highlow;
   INFILE 'c:\MyRawData\Temperature.dat';
   INPUT City $ State $
         / NormalHigh NormalLow
         #3 RecordHigh RecordLow;
PROC PRINT DATA = highlow;
   TITLE 'High and Low Temperatures for July';
RUN;
```

Input 后面告诉 SAS 读取第一行的 city 变量和 state 变量，斜线/告诉 SAS 移动到下一行的第一列，以便读取 normalhigh 和 normallow。#3 告诉 SAS 移动到第三行的第一列以便继续读取观测值的 recordhigh 变量和 recordlow 变量。这里/可以用#2 代替，也可以用/代替#3。

日志记录如下：

```
NOTE: 9 records were read from the infile 'c:\MyRawData\Temperature.dat'.
      The minimum record length was 5.
      The maximum record length was 10.

NOTE: The data set WORK.HIGHLOW has 3 observations and 6 variables.
```

从日志中可以看出，虽然原始原件占了 9 行，但只有三个观测值。

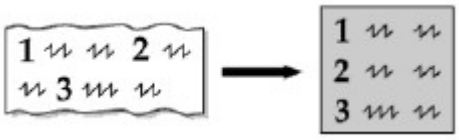
输出结果如下：

High and Low Temperatures for July 1

Obs	City	State	Normal High	Normal Low	Record High	Record Low
1	Nome	AK	55	44	88	29
2	Miami	FL	90	75	97	65
3	Raleigh	NC	88	68	105	50

第二章 将你的数据放入 SAS（2.11-2.21）

2.12 一行有多个观测值的原始文件读取



当一行出现多个观测值时，可以在 input 语句结尾加一个停止符号@@

例子 有一个关于降水量的数据，precipitation.dat，文件包含城市名、州名、月平均降水量、月平均降水天数：

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

这个数据文件中，第一行包含了两个观测值，可以用@@的程序读取：

```
* Input more than one observation from each record;
DATA rainfall;
   INFILE 'c:\MyRawData\Precipitation.dat';
   INPUT City $ State $ NormalRain MeanDaysRain @@;
PROC PRINT DATA = rainfall;
   TITLE 'Normal Total Precipitation and';
   TITLE2 'Mean Days with Precipitation for July';
RUN;
```

日志记录如下：

```
NOTE: 2 records were read from the infile 'c:\MyRawData\Precipitation.dat'
      The minimum record length was 18.
      The maximum record length was 28.

NOTE: SAS went to a new line when INPUT statement reached past the
      end of a line.

NOTE: The data set WORK.RAINFALL has 3 observations and
      4 variables.
```

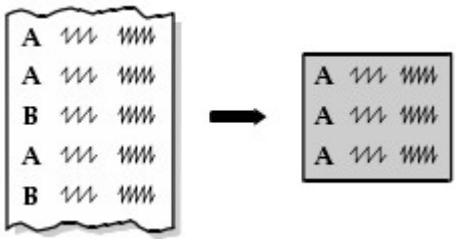
中间的说明，SAS went to a new line when INPUT statement reached past the end of a line.是指读取第二个值时达到第一行末尾，并转到下一行继续读取。通常这些信息会预示一个问题出现，但在这里它们都是你所想要的（为什么？）

输出结果如下：

Normal Total Precipitation and 1
Mean Days with Precipitation for July

Obs	City	State	Normal Rain	Mean DaysRain
1	Nome	AK	2.50	15
2	Miami	FL	6.75	18
3	Raleigh	NC	.	12

2.13 读取原始数据的部分观测值



有时候只需要读取原始数据的部分观测值，比如只需要年鉴中的女性数据、收入超过 10 万的人口数据等。

此时的数据读取方式如下：在 SAS 读取某一行观测值时，首先读取足够的变量以便决定是否需要保留此行的观测值。然后在 input 语句结尾加符号@，叫做 a trailing at（called a trailing at），这告诉 SAS 先停在（hold）此行，同时用 IF 语句检测此观测值是否满足需要，如果是，那么可以再用一个 input 语句来读取现有的变量。

例子 有一个关于当地交通的数据，traffic.dat 数据包含街道的类型（freeways 和 surface）、街道类型、早晨每小时的机动车流动量、晚上每小时机动车流动量。

```
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
```

如果现在你只需要 freeway 的数据，可以用下述程序：

```
* Use a trailing @, then delete surface streets;
DATA freeways;
   INFILE 'c:\MyRawData\Traffic.dat';
   INPUT Type $ @;
   IF Type = 'surface' THEN DELETE;
   INPUT Name $ 9-38 AMTraffic PMTraffic;
PROC PRINT DATA = freeways;
   TITLE 'Traffic for Freeways';
RUN;
```

第一个 input 读取字符串变量，@是 SAS 停留在观测值上并用 IF 检测，第二个 input 读取 input 后面的变量值。

程序执行后日志包括两部分说明，一个说明读取了 8 个记录，另一个说明新数据集中只包含三个观测值。

```
NOTE: 8 records were read from the infile 'c:\MyRawData\Traffic.dat'.
      The minimum record length was 47.
      The maximum record length was 47.

NOTE: The data set WORK.FREEWAYS has 3 observations and 4 variables.
```

输入结果如下所示：

Traffic for Freeways 1

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

@ vs @@ @的作用类似于@@，都是行停留指示符（line-hold specifiers），不同地方在于停留多久，@能使 SAS 停留到下一个 input 语句（也不换行），@@能使停留的时间到下一个 data 步（也不换行）。

比如这段代码：

```
data test;
    infile cards ;
    input x @;
    input y;
    input z @@;
cards;
1 2 3 4 5 6
7 8 9 10 11 12
13 14 15 16 17
;
run;
```

test 输出结果就是：

Obs	x	y	z
1	1	2	7
2	8	9	13

2.14 用 infile 语句中的选项控制输入

读取原始数据时，SAS 做了某些假设，比如从第一行开始读取数据，对于跨行观测值，会自动转到下一行继续读取。但有的特殊数据不满足这些假设，infile 语句中的选项可以让 SAS 读取这些特殊数据。

FIRSTOBS= FIRSTOBS= 选项告诉 SAS 从哪一行开始读取数据，当数据开头有些说明信息，或者想要跳过某些行时，这个选项很有用。例如，如下原始数据文件中，开头两行是关于数据的描述：

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

那么用如下程序可以让 SAS 从第三行开始读取数据：

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

OBS= OBS=告诉 SAS 一直读取到哪一行位置，注意是行而不是观测值（有的观测值占据多行）比如，如下的原始数据文件中，结尾处还有一句不需要的数据说明时。就需要这个选项：

```
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
```

用 FIRSTOBS=3 和 OBS=5 就可以读取第三行到第五行的数据：

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

MISSOVER 在 input 语句中输入的几个变量，SAS 在观测值中就读取几个变量，如果一行未读完，则进入下一行直到输入的变量都读取了变量值。missover 可以让 SAS 不进入下一行读取，未赋值的变量就使其成为缺失值。当如下这种数据，就需要 missover 选项，一个学生应该有 5 门课的成绩，但由于最后两门是自学课程，不是所有学生都完成，故而缺失：

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

如下的程序可以让 SAS 将 Nguyen 第五门课的成绩设为缺失值，从而不牵扯到下一行：

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

Truncover 使用 column input 或 formatted input 输入时可能会需要这个选项，因为这时有的数据行比其他的短。如下的原始数据中，由于三行的长度都不一样，input 中只能指定最长的一行：

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

程序如下：

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

这里指定了第二行的长度 street $ 22-37，但是第一行 maple ave.并没占够至第 37 列（注意后面是没有空格的），故而必须用 truncover，否则会转到下一行继续读取，第三行情况也是。

2.15 用数据步读取分隔符文件（delimited files）

分隔符文件中，变量值之间会用一些特殊的字符隔开，比如逗号或制表符。DLM=和 DSD 选项可以让 SAS 容易的读取这些分隔符文件。

DLM= 用 list input 读取文件时，变量值之间应该用空格隔开。对于其他的分隔符，可以用 DLM=，DELIMITER=选项来指定，从而可以读取文件。

例子 如下的数据中，学生姓名、每周读的书的数目是用逗号隔开的：

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

用选项来指定分隔符即可：

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

如果原始数据是用制表符隔开的，那么可以使用 DLM='09'X 来指定，因为制表符的十六进制值是 09，如果你电脑使用 EBCDIC（扩充的二进制编码的十进制交换码），那么应该用 DLM='05'X。

DSD DSD (Delimiter-Sensitive Data)有三个作用：忽略引号中数值的分隔符；自动将字符数据中的引号去掉；将两个相邻的分隔符当做缺失值来处理。并且，DSD 默认分隔符为逗号，如果数据中的分隔符不是逗号，那么要用 delimiter 来指定。比如，读取一个制表符为分隔符、并且用两个制表符代表缺失值的数据文件，则要用下面的语句：

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

CSV 文件 CSV 文件，Comma-separated values files，是可以用 DSD 选项的文件类型。Excel 可以储存 CSV 格式的文件。

例子 某咖啡馆，老板每晚请不同的乐队表演来吸引顾客，他记录了乐队名称、演出日期、

晚上 8 点、9 点、10 点、11 点的顾客数量：

```
Lupine Lights,12/3/2003,45,63,70,
Awesome Octaves,12/15/2003,17,28,44,12
"Stop, Drop, and Rock-N-Roll",1/5/2004,34,62,77,91
The Silveyville Jazz Quartet,1/18/2004,38,30,42,43
Catalina Converts,1/31/2004,56,,65,34
```

注意，其中有一个乐队的名字中用逗号来分隔，并且使用了引号。最后一条记录中还有一个缺失值，用两个连续的逗号表示。INFILE 语句中的 DSD 选项可以用来读取这个文件，并且，由于每个记录长度不一样，还需要用 missover：

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

注意 bandname 和 GigDate 两个变量使用了**冒号修改器**，冒号修改器告诉 SAS 读取信息的长度（BandName 为 30，GigDate 为 10）。输出结果如下：

Customers at Each Gig 1

Obs	BandName	Gig Date	Eight PM	Nine PM	Ten PM	Eleven PM
1	Lupine Lights	16042	45	63	70	.
2	Awesome Octaves	16054	17	28	44	12
3	Stop, Drop, and Rock-N-Roll	16075	34	62	77	91
4	The Silveyville Jazz Quartet	16088	38	30	42	43
5	Catalina Converts	16101	56	.	65	34

2.16 用导入过程（IMPORT procedure）读取分隔符文件。

Proc import 会浏览你的数据文件，自动决定变量类型（字符串或数值），为字符串变量分配正确的长度，辨认出日期变量。Proc import 会将两个连续的分隔符视为缺失值，会读取引号中的变量值。一行读完后，会自动分配缺失值给未赋值的变量。Also,if you want,you can use the first line in your data file for the variable names。导入过程（IMPORT procedure）自动问你写下数据步，这可以在提交之后的日志窗口中查看。

一个导入过程（IMPORT procedure）的最简单形式：

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

用语句 DATAFILE='filename'读取文件名，用 OUT=data-set 创建 SAS 数据集。SAS 会通过文件的扩展名来检测文件的类型：

Type of File	Extension	DBMS Identifier
Comma-delimited	.csv	CSV
Tab-delimited	.txt	TAB
Delimiters other than commas or tabs		DLM

如果文件没有正确的扩展名，或者是 DLM 格式的，必须在 proc import 语句中用 DBMS=option。如果想要创建的数据集名字已经存在，那么要用 replace 选项代替。一个使用 replace 和 dbms 的例子。

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

导入过程（IMPORT procedure）从数据文件中的第一行获取变量名，可以通过在 PROC IMPORT 后面增加 GETNAMES=NO 语句来改变这种默认，PROC IMPORT 会分配给变量名字：VAR1，VAR2，VAR3 等。如果你的数据文件是 DLM 类型的，PROC IMPORT 会假定分隔符为空格，用 DELIMITER=可以改变默认的分隔符。如下是一段有上述代码的程序：

```
PROC IMPORT DATAFILE = 'filename' OUT = data-set
     DBMS = DLM REPLACE;
   GETNAMES = NO;
   DELIMITER = 'delimiter-character';
RUN;
```

例子 下面还是使用咖啡馆中，乐队表演的例子（2.15），注意其中有一个乐队的名字中用逗号来分隔，并且使用了引号：

```
Lupine Lights,12/3/2003,45,63,70,
Awesome Octaves,12/15/2003,17,28,44,12
"Stop, Drop, and Rock-N-Roll",1/5/2004,34,62,77,91
The Silveyville Jazz Quartet,1/18/2004,38,30,42,43
Catalina Converts,1/31/2004,56,,65,34
```

用 proc import 读取数据的代码如下：

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

输出结果如下，注意 GigDate 的日期格式能够被 proc import 辨认出来：

Customers at Each Gig 1

Obs	Band_Name	Gig_Date	Eight_PM
1	Lupine Lights	12/03/2003	45
2	Awesome Octaves	12/15/2003	17
3	Stop, Drop, and Rock-N-Roll	01/05/2004	34
4	The Silveyville Jazz Quartet	01/18/2004	38
5	Catalina Converts	01/31/2004	56

Obs	Nine_PM	Ten_PM	Eleven_PM
1	63	70	.
2	28	44	12
3	62	77	91
4	30	42	43
5	.	65	34

2.17 用导入过程（IMPORT procedure）读取 PC 文件

如果安装了 SAS/ACCESS 模块，导入过程（IMPORT procedure）可以导入一些 PC 文件类型。它会浏览你的文件以决定变量类型，并默认使用数据的第一行来分配变量名。Windows 操作环境中可以导入 excel、Lotus、dBase、和 Access 文件。Unix 系统中可以导入 dBase 文件，并且从 SAS9.1 开始，Unix 系统也可以导入 excel 和 access 文件。在 windows 环境中有一个不需要 SAS/ACCESS 模块的方法——Dynamic Data Exchange(DDE)，将在 2.18 中讲解。

Microsoft Excel，Lotus，和 dBase 文件 下面是用导入过程（IMPORT procedure）读取 PC 文件的一般过程：

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

如果读取的文件是如下类型，就不用 DBMS=OPTION。

Type of File	Extension	DBMS Identifier
Microsoft Excel	.xls	EXCEL[3]
		EXCEL5
		EXCEL4
Lotus	.wk4	WK4
	.wk3	WK3
	.wk1	WK1
dBase	.dbf	DBF

在读取 excel 时，有时需要指定要读取的是哪一个工作薄——sheet

SHEET=name-of-sheet;

默认情况下，导入过程（IMPORT procedure）会从工作薄的第一行中读取变量名。如果不需要，可以用如下代码使得 SAS 给变量赋名为 F1，F2 等。

GETNAMES=NO;

Microsoft Access Files 读取这种文件需要用 DATABASE= 和 DATATABLE=，而不是 DATAFILE=option。

```
PROC IMPORT DATABASE = 'database-path' DATATABLE = 'table-name'
     OUT = data-set DBMS = identifier  REPLACE;
```

下面的是 DBMS 可以辨认的 Access 文件

Type of File	Extension	DBMS Identifier
Microsoft Access	.mdb	ACCESS[5] ACCESS97

例子 有如下的 EXCEL 数据：

	A	B	C	D	E	F	G
1	Visiting Team	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

读取的 proc import 程序：

```
* Read an Excel spreadsheet using PROC IMPORT;
PROC IMPORT DATAFILE = 'c:\MyExcelFiles\Onions.xls' OUT = sales;
PROC PRINT DATA = sales;
   TITLE 'SAS Data Set Read From Excel File';
RUN;
```

输出结果如下：

SAS Data Set Read From Excel File 1

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 用 DDE 读取 PC 文件

DDE，动态数据交换（Dynamic Data Exchange），读取 PC 文件的优点为：可以直接访问存于 PC 文件中的数据，不要求购买其他 SAS 产品；缺点为：只能用在 windows 环境下，只能在程序运行时（比如 excel），SAS 才能进行读取。

有几种方式可以用 DDE 访问数据：

- 复制数据到剪贴板
- 指定 DDE 三元组
- 从 SAS 中启动 PC 程序，然后读取数据。

复制数据到剪贴板 可以直接复制数据至剪贴板，然后再 SAS 程序的 DDE FILENAME 语句中是使用 CLIPBOARD 关键字。比如，excel 中有如下的工作薄：

	A	B	C	D	E	F	G
1	Visiting Team	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

复制 A2 到 G5，然后在不关闭 excel 的状态下，提交如下 SAS 程序：

```
* Read an Excel spreadsheet using DDE;
FILENAME baseball DDE 'CLIPBOARD';
DATA sales;
  INFILE baseball NOTAB DLM='09'x DSD MISSOVER;
  LENGTH VisitingTeam $ 20;
  INPUT VisitingTeam CSales BSales OurHits TheirHits OurRuns TheirRuns;
RUN;
```

FILENAME 语句将指代的文件（BASEBALL）定义成 DDE 类型，并指定从剪贴板中去读取它（CLIPBOARD）。DDE 默认空格为分隔符，如果变量值之间有空格，则要在 INFILE 语句中用 NOTAB 选项和 DLM='09'X 选项，前者告诉 SAS 在变量值之间放置制表符，后者告诉 SAS 将制表符定义为分隔符。如果数据中有缺失值，则要在 INFILE 中加入 DSD 和 MISSOVER 选项，前者将两个连续的分隔符视为缺失值，后者告诉 SAS 如果此行读完，不要进入下一行给未赋值的变量赋值。

指定 DDE 三元组 这种方法可以不用复制数据，直接指定出文件的 DDE 三元组。DDE 三元组的形式为：**application| topic ! item**。

有一种方法可以在 SAS 中直接查看文件的 DDE 三元组，方法为：复制数据至剪贴板里，触发 SAS 会话，从解决方案（Solution）菜单中选择附件（accessories）——DDE 三元组。一个窗口会出现你复制文件的 DDE 三元组。比如，一个工作薄的 DDE 三元组为：

Excel|C:\MyFiles\[BaseBall.xls]sheet1!R2C1:R5C7

读取这个文件的 FILENAME 语句为：

FILENAME baseball DDE'Excel|C:\MyFiles\[BaseBall.xls]sheet1!R2C1:R5C7';

从 SAS 中启动程序 这种方法可以不用在运行 SAS 之前启动数据程序。想要从 SAS 中启动程序，然后读取数据，则首先需要 NOXWAIT 和 NOXSYNC 系统选项，然后使用 X 语句，一个例子：

```
* Read an Excel spreadsheet using DDE;
OPTIONS NOXSYNC NOXWAIT;
X '"C:\MyFiles\BaseBall.xls"';
FILENAME baseball DDE 'Excel|C:\MyFiles\[BaseBall.xls]sheet1!R2C1:R5C7';
DATA sales;
  INFILE baseball NOTAB DLM='09'x DSD MISSOVER;
  LENGTH VisitingTeam $ 20;
  INPUT VisitingTeam CSales BSales OurHits TheirHits OurRuns TheirRuns;
RUN;
```

NOXSYNC 和 NOXWAIT 语句告诉 SAS 不要等待用户输入。X 语句告诉 windows 执行或打开引号中路径的文件，注意这里路径设置了两个引号，如果路径中有空格，则要设置两个引号。使用这种方法，必须要在 FILENAME 语句中指定 DDE 三元组。

2.19 临时和永久数据集

SAS 临时数据集只在目前工作或会话中存在，关闭 SAS 或结束工作时则删除；永久数据集当关闭 SAS 或结束工作时仍然存在。

SAS 数据集名 所有的 SAS 数据集都有用句号分开的两层数据集名，如 work.a。第一层前缀 work 是逻辑库名，第二层是在逻辑库中用于辨别自己的成员名。

名字的规则是，以字母或下划线开头，并且名字中只能包含字母、数字和下划线。而且，库名不能超过 8 个字节，而成员名却可以达到 32 个字节。

大部分数据集通过数据步创建，过程步也可以创建。如果指定了一个前缀不为 work 的两层数据集名，则这个数据集就是永久的。如果不指定前缀，则默认数据集是临时的，自动分配到 work 逻辑库中。下面是一些数据集名，对于的逻辑库，成员名，类型：

DATA statement	Libref	Member name	Type
DATA ironman;	WORK	ironman	temporary
DATA WORK.tourdefrance;	WORK	tourdefrance	temporary
DATA Mylib.doublecentury;	Mylib	doublecentury	permanent

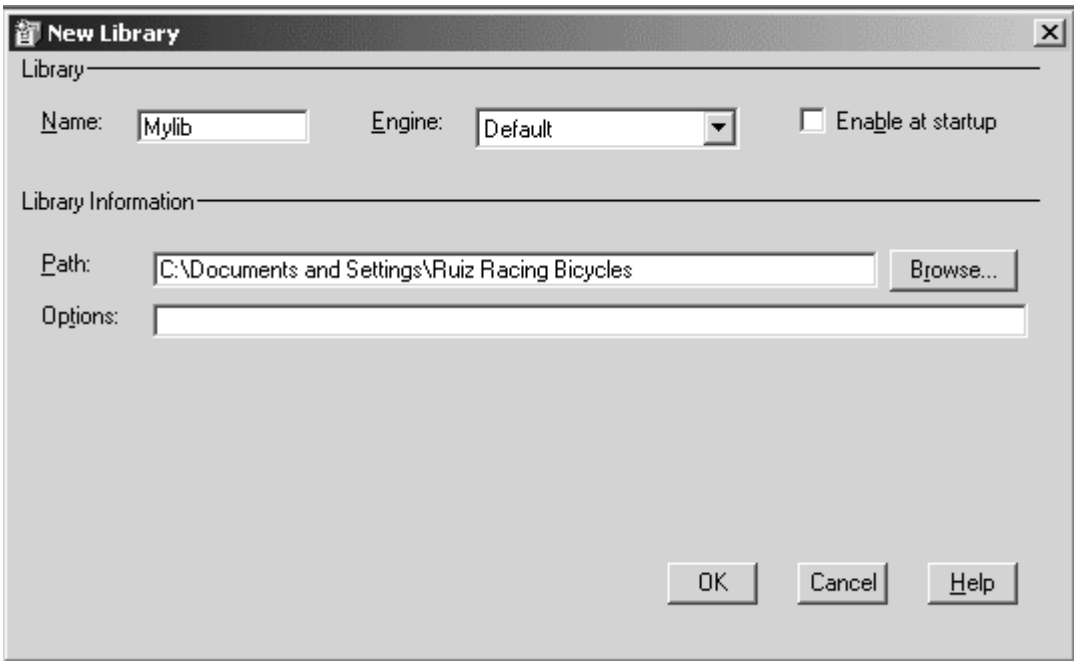
临时数据集 如下的程序创建并打印了一个名为 DISTANCE 的永久数据集：

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

这里，只指定了成员名 distance，自动分配到 work 库中，日志窗口中有说明：

NOTE:The data set **WORK.DISTANCE** has 1 observations and 2 variables.

永久数据集 可以在资源管理器窗口中定义一个新库使用：



也可以通过如下程序：

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

那么日志窗口就会出现如下说明：

NOTE:The data set **MYLIB.DISTANCE** has 1 observations and 2 variables.

这是一个永久数据集，因为前缀不是 work。

2.20 用 LIBNAME 语句使用永久数据集

LIBNAME 语句的基本形式为：LIBNAME libref'your-SAS-data-library';

LIBNAME 的后面，需要指定库名和存放的路径，在个人操作环境下 LIBNAME 语句的基本形式为：

```
Windows:          LIBNAME libref 'drive:\directory';
UNIX:             LIBNAME libref '/home/path';
OpenVMS:          LIBNAME libref '[userid.directory]';
OS/390 or z/OS:   LIBNAME libref 'data-set-name';
```

创建永久数据集 如下的例子创建了一个永久 SAS 数据集，包含了 magnolia trees 的一些信息。每一种树，原始文件都包含它的科学名、普通名、最大高度、第一次开花的年龄、是 evergreen 还是 deciduous、以及花的颜色。

```
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
```

下面的代码将会创建一个 PLANTS 的逻辑库，路径为 C 盘下的 MySASLib。然后从原始文件 Mag.dat 中读取数据，并创建一个名为 MAGNOLIA 的永久数据集，存在 PLANTS 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;
```

日志窗口会出现如下说明：

NOTE:The data set **PLANTS.MAGNOLIA** has 5 observations and 6 variables.

如果在电脑中打印文件的地址目录，会发现文件名不是 PLANTS.MAGNOLIA。这是因为操作系统有自己对文件命名的方式，这个文件，在 Windows,UNIX,和 OpenVMS 操作环境中名字为 magnolia.sas7bdat，在 OS/390 或者 z/OS 环境中，文件名就会如 LIBNAME 语句中定义的 data-set-name 形式。

读取永久数据集 如果你想打印出上例中创建的数据集，可以用如下语句：

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

这次 LIBNAME 语句中的库名为 example，但缺失同样路径，逻辑库名可以改变，但成员名 MAGNOLIA 却一样。输出如下：

Magnolias 1

Obs	ScientificName	CommonName	Maximum Height	Age Bloom	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 通过直接指代使用永久数据集

可以通过直接指代来使用 SAS 数据集，且不需要自己定义，SAS 为你做好。

直接指代，依据系统不同，使用方法也不同，如下:

```
Windows:        DATA 'drive:\directory\filename';
UNIX:           DATA '/home/path/filename';
OpenVMS:        DATA '[userid.directory]filename';
OS/390 or z/OS: DATA 'data-set-name';
```

可以看到，一些系统的语句中需要指出路径，但如果遗漏了路径，SAS 自动使用当前路径，比如这样一个创建名为 trees 的永久数据集的代码：

DATA ‘trees’;

UNIX 和 OPENVMS 操作环境下，当前的路径默认为启动 SAS 的路径，可以通过工具（TOOLS）下拉菜单的选项（OPTIOPN）菜单来改变这种默认，windows 环境下当前路径会显示在 SAS 窗口底部。可以通过双击这个路径来改变默认。

例子 如下还是关于 magnolia trees 的这个例子，

```
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
```

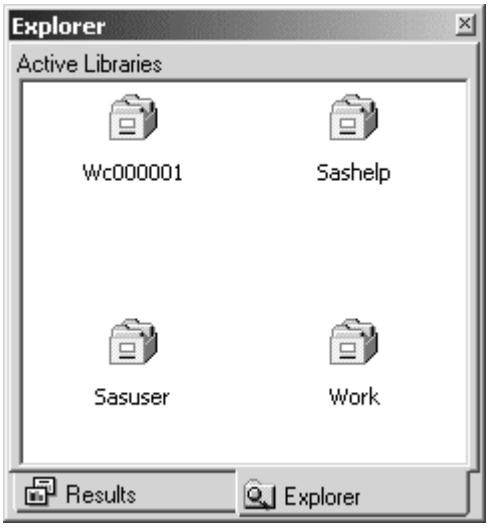
下面的代码将从原始文件 mag.dat 中读取数据，创建一个名为 MAGNOLIA 的永久数据集，存放在 C 盘的 Mysaslib 路径中：

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

相应的输出窗口显示如下：

NOTE:The data set c:\MySASLib\magnolia has 5 observations and 6 variables.

如果打开 MySASLib 文件夹，会发现一个名为 magnolia.sas7bdat 的文件。在没指定库的情况下，SAS 会自动为你创建一个库，在资源管理器窗口中可以看到，下图是 SAS 为 magnolia 创建的库。



用直接指代读取 SAS 数据集 可以直接用引号+路径的方式读取永久数据集，比如打印 magnolia 数据集可以：

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

输出窗口如下：

Magnolias 1

Obs	ScientificName	CommonName	Maximum Height	Age Bloom	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

2.22 列出 SAS 数据集目录

由于 SAS 是自文档化，即在自动储存了数据集的信息，因此可以通过 contents 过程来查看 SAS 数据集的描述。

Proc contents data=data-set

如果遗漏了 data=的语句，SAS 自动列出最近创建的数据集

例子 如下的程序创建了一个数据集，并且使用 proc contents。数据步中使用了 label 语句，label 语句为变量打上标签，并储存在数据集中，在打印时会显示。过程步中也可以使用 label，但只在 proc contents 中有效，不会储存在数据集中。Informat 和 format 可以指定信息和格式，储存在数据集中，也可以在过程步中使用，但不储存在数据集中。

```
DATA funnies;
   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
   ;
* Use PROC CONTENTS to describe data set funnies;
PROC CONTENTS DATA = funnies;
RUN;
```

输出如下：

```
                        The CONTENTS Procedure

❶ Data Set Name      WORK.FUNNIES                ❷ Observations          4
  Member Type        DATA                        ❸ Variables             5
  Engine             V9                          Indexes                 0
❹ Created            13:36 Monday, May 12, 2003  Observation Length     40
  Last Modified      13:36 Monday, May 12, 2003  Deleted Observations    0
  Protection                                     Compressed              NO
  Data Set Type                                  Sorted                  NO
  Label
  Data Representation WINDOWS
  Encoding wlatin1    wlatin1 Western (Windows)

              -----Engine/Host Dependent Information-----
  Data Set Page Size          4096
  Number of Data Set Pages    1
  First Data Page             1
  Max Obs per Page            101
  Obs in First Data Page      4
  Number of Data Set Repairs  0
  File Name                   C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\SAS
                              Temporary Files\_TD832\funnies.sas7bdat
  Release Created             9.0000M0
  Host Created                XP_HOME

          -----Alphabetic List of Variables and Attributes-----
  #   Variable ❶Type ❷Len   ❸Format      ❹Informat ❺Label

  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
```

第三章 开发你的数据

3.1 创建并重新定义变量

可以通过分配语句来创建并重新定义变量，基本形式为：

Variable=expression

Variable 是变量名，expression 可以是常量、其他变量、或者数学表达式。分配语句的基本类型有：

Type of expression	Assignment statement
numeric constant	`Qwerty = 10;`
character constant	`Qwerty = 'ten';`
a variable	`Qwerty = OldVar;`
addition	`Qwerty = OldVar + 10;`
subtraction	`Qwerty = OldVar - 10;`
multiplication	`Qwerty = OldVar * 10;`
division	`Qwerty = OldVar / 10;`
exponentiation	`Qwerty = OldVar ** 10;`

Expression 是数学表达式时，需要遵循运算法则，先算指数、再算乘除、之后是加减。可以用括号改变运算等级。

例子 如下是一个农产品估重数据，每位农民要求对他们的番茄、南瓜、豌豆、葡萄进行估重：

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

下面代码从 garden.dat 原始文件中读取数据，并进行修改：

```
* Modify homegarden data set with assignment statements;
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;
PROC PRINT DATA = homegarden;
   TITLE 'Home Gardening Survey';
RUN;
```

这个程序包含了 5 句分配语句，第一个将 14 赋值给 zone，第二个使 type 等于一个字符串常量……打印出的结果中，既包括旧变量，又包括新变量：

Home Gardening Survey 1

Obs	Name	Tomato	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	.	.

由于观测值 susan 的 peas 变量出现了缺失值，因此这个观测值的 total 和 pertom 变量也出现了缺失值。日志窗口的说明如下：

NOTE:Missing values were generated as a result of performing an operation onmissing values.

3.2 使用 SAS 函数

SAS 有 400 多个函数，主要涵盖如下领域:

Character	Probability
Date and Time	Random Number
Financial	Sample Statistics
Macro	State and ZIP Code
Mathematical	

函数基本形式：function-name(argument,argument,…)，所有的函数都需要括号，即使没有参数。下面的程序计算使用 MDY 函数，MDY 函数需要三个参数：月、日、年。

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

函数可以嵌套，即一个函数可以是另一个函数的参数。比如：NewValue=INT(LOG(10));

例子 有一个南瓜雕刻比赛的数据，pumpkin.dat 数据包含了参赛者的名字、年龄、雕刻的南瓜类型、报名日期、五位裁判给出的分数。

```
Alicia Grossman  13 c 10-28-2003 7.8 6.5 7.2 8.0 7.9
Matthew Lee       9 D 10-30-2003 6.5 5.9 6.8 6.0 8.1
Elizabeth Garcia 10 C 10-29-2003 8.9 7.9 8.5 9.0 8.8
Lori Newcombe     6 D 10-30-2003 6.7 5.6 4.9 5.2 6.1
Jose Martinez     7 d 10-31-2003 8.9 9.510.0 9.7 9.0
Brian Williams   11 C 10-29-2003 7.8 8.4 8.5 7.9 8.0
```

下面的代码读取了数据、创建了两个新变量、转换了一个大小写：

```
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);
PROC PRINT DATA = contest;
   TITLE 'Pumpkin Carving Contest';
RUN;
```

AvgScore 使用均值函数创建的变量，计算参数的均值，这与直接相加再除以 5 不同的地方在于，当参数中出现缺失值时，直接相加再除的方法返回缺失值，而均值函数计算非缺失参数的均值。

DayEntered 变量使用 DAY 函数，返回日期在一个月里的天数。

Type 用大写转换函数将原来的字母转换成大写字母。

结果是：

Pumpkin Carving Contest 1

Obs	Name	Age	Type	Date'	Scr1	Scr2	Scr3	Scr4	Scr5	Avg Score	Day Entered
1	Alicia Grossman	13	C	16006	7.8	6.5	7.2	8.0	7.9	7.48	28
2	Matthew Lee	9	D	16008	6.5	5.9	6.8	6.0	8.1	6.66	30
3	Elizabeth Garcia	10	C	16007	8.9	7.9	8.5	9.0	8.8	8.62	29
4	Lori Newcombe	6	D	16008	6.7	5.6	4.9	5.2	6.1	5.70	30
5	Jose Martinez	7	D	16009	8.9	9.5	10.0	9.7	9.0	9.42	31
6	Brian Williams	11	C	16007	7.8	8.4	8.5	7.9	8.0	8.12	29

3.3 选择 SAS 函数

3.4 使用 IF-THEN 语句

条件语句 IF-THEN 的基本形式为：IF 条件 THEN 执行;

比如：IF Model='Mustang' THEN Make='Ford';

条件语句中的一些基本比较符号：

Symbolic	Mnemonic	Meaning
=	EQ	equals
¬ =, ^ =, or ~ =	NE	not equal
>	GT	greater than
<	LT	less than
> =	GE	greater than or equal
< =	LE	less than or equal

还有 IN 比较符，比如这句中 IF Model IN('Corvette','Camaro') THEN Make='Chevrolet';代表当 Model 为 Corvette 或 Camaro 的时候，将 Chevrolet 赋给 Make。

一个条件只能有一个执行，如果要多个执行，则需要 DO 和 END 关键字。

```
IF condition THEN DO;
   action;
   action;
END;
```

```
IF Model = 'Mustang' THEN DO;
   Make = 'Ford';
   Size = 'compact';
END;
```

可以用 AND 和 OR 来定义多个条件：IF Model='Mustang' AND Year<1975 THEN Status='classic';

Symbolic	Mnemonic	Meaning
&	AND	all comparisons must be true
|, !, or !	OR	only one comparison must be true

例子 如下的数据包含了模型的名字、年份、制造商和颜色：

```
Corvette 1955 .      2 black
XJ6      1995 Jaguar 2 teal
Mustang  1966 Ford   4 red
Miata    2002 .      . silver
CRX      2001 Honda  2 black
Camaro   2000 .      4 red
```

下面的代码从 cars.dat 的原始文件中读取数据，使用 IF-THEN 语句填满缺失值，并创建一个新变量 Status

```
DATA sportscars;
   INFILE 'c:\MyRawData\Cars.dat';
   INPUT Model $ Year Make $ Seats Color $;
   IF Year < 1975 THEN Status = 'classic';
   IF Model = 'Corvette' OR Model = 'Camaro' THEN Make = 'Chevy';
   IF Model = 'Miata' THEN DO;
      Make = 'Mazda';
      Seats = 2;
   END;
PROC PRINT DATA = sportscars;
   TITLE "Eddy's Excellent Emporium of Used Sports Cars";
RUN;
```

输出结果如下：

Eddy's Excellent Emporium of Used Sports Cars 1

Obs	Model	Year	Make	Seats	Color	Status
1	Corvette	1955	Chevy	2	black	classic
2	XJ6	1995	Jaguar	2	teal	
3	Mustang	1966	Ford	4	red	classic
4	Miata	2002	Mazda	2	silver	
5	CRX	2001	Honda	2	black	
6	Camaro	2000	Chevy	4	red	

3.5 用 IF-THEN 语句将观测值分组

IF THEN/ELSE 的一般形式为：

IF condition THEN action;

ELSE IF condition THEN action;

ELSE IF condition THEN action;

用 else 语句与直接用多个 IF-THEN 语句比起来，有两个优势，第一是更有效率，电脑将占用更少的时间；第二是 else 可以确保你的两个 condition 之间互斥。

有时候最后一个 ELSE 只有 action，没有 IF-THEN：

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

例子 有一个住房改善的数据，home.dat，包括了姓名、改善工作、改善成本：

```
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
```

下面的代码读取数据，并新建了一个 CostGroup 的变量。根据 Cost 的值将数据分成 high、medium、low 和 missing 三类：

```
* Group observations by cost;
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';
PROC PRINT DATA = homeimprovements;
   TITLE 'Home Improvement Cost Groups';
RUN;
```

输出结果是：

Home Improvement Cost Groups 1

Obs	Owner	Description	Cost	Cost Group
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

3.6 构造子集

IF 语句可以构造子集，取数据集中的部分数据。

基本形式为： IF expression; 比如： IF Sex='f';

如果 IF 条件中的数据是真，则数据步将继续执行。

还可以使用 DELETE 语句，来删除哪些不要的数据： IF expression THEN DELETE;

这两句话是等价的： IF Sex='f'; IF Sex='m'THEN DELETE;

例子 有关于莎士比亚歌剧的清单，Shakespeare.dat，包含歌剧名、首次表演年份、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
```

下面的代码读取数据，并且用 IF 语句构造一个只包含喜剧（comedies）的子集：

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

输出结果如下：

Shakespearean Comedies 1

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

观察日志有时能很好的保证我们截取了我们要的数据：

```
NOTE: 8 records were read from the infile 'c:\MyRawData\Shakespeare.dat'
NOTE: The data set WORK.COMEDY has 3 observations and 3 variables.
```

在这个例子中，用 DELETE 等价的语句为：

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

3.7 处理 SAS 的日期数据

日期数据的处理很棘手，有的月份有 31 天、有的 30 天、有的 28 天。SAS 简化的日期数据，将所有的日期转化成一个以 1960 年 1 月 1 日为起点的数。比如：

Date	SAS date value
January 1, 1959	-365
January 1, 1960	0
January 1, 1961	366
January 1, 2003	15706

SAS 处理日期数据的三个工具为：读取数据的 informats，使用数据的函数（functions），打印数据的 formats

Informats 读取日期数据需要用 formatted input。比如，如何告诉 SAS 用 MMDDYY10. imforat 读取名为 BirthDate 的变量：

INPUT BirthDate MMDDYY10.;

设定默认的百年 07/04/76 这样的数据可能是 1976，也可能是 2076、1776。因此需要 YEARCUTOFF=来指定一个一百年的第一年，默认的是 1920 年。下面的语句就是告诉 SAS 将一个两位年份的日期解释为 1960 年到 2049 年之间：

OPTIONS YEARCUTOFF=1950;

SAS 表达式中的日期 一旦被以 SAS 日期格式读取之后，可以将此数据想其他数值数据一样用在表达式中。比如像为图书馆的书设定 21 天的还书日期，只需要在结束日期上加上 21：

OPTIONS YEARCUTOFF=1950;

通过在表达式中加入引号和字母 D，可以将一个日期当做常数来使用，如下的代码创建了一个 EarthDay05 的变量，并且等于 April 22,2005

EarthDay05='22APR2005'D;

函数 SAS 日期函数使得操作大大简便，比图 today（）返回今天的日期。

语句 DaysOverDue=TODAY()-DateDue;可以计算一本书应归还还剩余的期限。

Fomats 打印日期数据时，还需要将数值换成日期，下面的 FORMAT 语句告诉 SAS 用 WEEKDATE17.格式打印变量 BirthDate。

FORMAT BirthDate WEEKDATE17.;

例子 图书馆有借书卡数据，Dates.dat，包含持卡人姓名、出生日期、卡办理日期。

```
A. Jones     1jan60     9-15-03
M. Rincon    05OCT1949 02-29-2000
Z. Grandage  18mar1988 10-10-2002
K. Kaminaka  29may2001 01-24-2003
```

下面的代码读取数据，计算变量使用期限（expiredate），使用期限为 3 年；变量 expirequarter 计算使用期限的四分之一，使用函数 QTR（）。接着用 IF 语句来判断一个卡是否为新卡，在 2003 年 1 月 1 日之后办理的，为新卡：

```
DATA librarycards;
   INFILE 'c:\MyRawData\Dates.dat' TRUNCOVER;
   INPUT Name $11. +1 BirthDate DATE9. +1 IssueDate MMDDYY10.;
   ExpireDate = IssueDate + (365.25 * 3);
   ExpireQuarter = QTR(ExpireDate);
   IF IssueDate > '01JAN2003'D THEN NewCard = 'yes';
PROC PRINT DATA = librarycards;
   FORMAT IssueDate MMDDYY8. ExpireDate WEEKDATE17.;
   TITLE 'SAS Dates without and with Formats';
RUN;
```

输出结果为：

SAS Dates without and with Formats 1

Obs	Name	Birth Date	Issue Date	ExpireDate	Expire Quarter	New Card
1	A. Jones	0	09/15/03	Thu, Sep 14, 2006	3	yes
2	M. Rincon	-3740	02/29/00	Fri, Feb 28, 2003	1	
3	Z. Grandage	10304	10/10/02	Sun, Oct 9, 2005	4	
4	K. Kaminaka	15124	01/24/03	Mon, Jan 23, 2006	1	yes

注意 BirthDate 没有用日期格式。

3.8 可选择的 Date Informats，Functions 和 Formats

Informats	Definition	Width range	Default width
DATE*w*.	Reads dates in form: *ddmmmyy* or *ddmmmyyyy*	7-32	7
DDMMYY*w*.	Reads dates in form: *ddmmyy* or *ddmmyyyy*	6-32	6
JULIAN*w*.	Reads Julian dates in form: *yyddd* or *yyyyddd*	5-32	5
MMDDYY*w*.	Reads dates in form: *mmddyy* or *mmddyyyy*	6-32	6

Functions	Syntax	Definition
DATEJUL	DATEJUL(*julian-date*)	Converts a Julian date to a SAS date value[2]
DAY	DAY(*date*)	Returns the day of the month from a SAS date value
MDY	MDY(*month,day,year*)	Returns a SAS date value from month, day, and year values
MONTH	MONTH(*date*)	Returns the month (1-12) from a SAS date value
QTR	QTR(*date*)	Returns the yearly quarter (1-4) from a SAS date value
TODAY	TODAY()	Returns the current date as a SAS date value

Formats	Definition	Width range	Default width
DATE*w.*	Writes SAS date values in form: *ddmmmyy*	5-9	7
DAY*w.*	Writes the day of the month from a SAS date value	2-32	2
EURDFDD*w.*	Writes SAS date values in form: *dd.mm.yy*	2-10	8
JULIAN*w.*	Writes a Julian date from a SAS date value	5-7	5
MMDDYY*w.*	Writes SAS date values in form: *mmddyy* or *mmddyyyy*	2-10	8
WEEKDATE*w.*	Writes SAS date values in form: *day-of-week, month-name dd, yy or yyyy*	3-37	29
WORDDATE*w.*	Writes SAS date values in form: *month-name dd, yyyy*	3-32	18

下面是例子：

Informats	Input data	INPUT statement	Results
DATE*w.*	1jan1961	`INPUT Day DATE10.;`	366
DDMMYY*w.*	01.01.61 02/01/61	`INPUT Day DDMMYY8.;`	366 367
JULIAN*w.*	61001	`INPUT Day JULIAN7.;`	366
MMDDYY*w.*	01-01-61	`INPUT Day MMDDYY8.;`	366

Functions	Example	Result	Example	Results
DATEJUL	`a=60001;` `x=DATEJUL(a);`	x=0	`a=60365;` `y=DATEJUL(a);`	y=364
DAY	`a=MDY(4,18,99);` `x=DAY(a);`	x=18	`a=MDY(9,3,60);` `y=DAY(a);`	y=3
MDY	`x=MDY(1,1,60);`	x=0	`m=2; d=1; y=60;` `Date=MDY(m,d,y);`	Date=31
MONTH	`a=MDY(4,18,1999)` `x=MONTH(a);`	x=4	`a=MDY(9,3,60);` `y=MONTH(a);`	y=9
QTR	`a=MDY(4,18,99);` `x=QTR(a);`	x=2	`a=MDY(9,3,60);` `y=QTR(a);`	y=3
TODAY	`x=TODAY();`	x=*today's date*	`x=TODAY()-1;`	x=*yesterday's date*

Formats	Input data	PUT statement[3]	Results
DATE*w*.	8966	PUT Birth DATE7.; PUT Birth DATE9.;	19JUL84 19JUL1984
DAY*w*.	8966	PUT Birth DAY2.; PUT Birth DAY7.;	19 19
EURDFDD*w*.	8966	PUT Birth EURDFDD8. PUT Birth EURDFDD10.;	19.07.84 19.07.1984
JULIAN*w*.	8966	PUT Birth JULIAN5.; PUT Birth JULIAN7.;	84201 1984201
MMDDYY*w*.	8966	PUT Birth MMDDYY8.; PUT Birth MMDDYY6.;	07/19/84 071984
WEEKDATE*w*.	8966	PUT Birth WEEKDATE15.; PUT Birth WEEKDATE29.;	Thu, Jul 19, 84 Thursday, July 19, 1984
WORDDATE*w*.	8966	PUT Birth WORDDATE12.; PUT Birth WORDDATE18.;	Jul 19, 1984 July 19, 1984

3.9 使用 retain 和 sum 语句

当开始数据步的每一个观测值迭代时，SAS 会先将所有变量值设为确实，再通过 input 和分配语句改变。Ratain 和 sum 语句可以改变这种方式，

Retain 语句 retain 语句可以让 SAS 保存前一次变量的值。它可以出现在数据步的任何位置，基本形式为：

RETAIN variable-list;

也可以指定一个初始值，而不是用缺失值或前一次的值代替初始值

RETAIN variable-list initial-value;

Sum 语句 SUM 语句用于你想将一个表达式的值累加到一个变量上去时，基本形式为：

variable+expression;

这个语句将表达式的值赋给变量，同时将变量的值保留到下一次迭代。这个变量必须是数值型，且初始值为 0。因此，语句等价于如下形式：

RETAIN variable 0;

variable=SUM(variable,expression);

例子 有一个关于本赛季棒球比赛的数据，games.dat，包含比赛日期、参赛队伍、hit 数据、runs 数据

```
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
```

现在需要增加两个变量，一个反应本赛季的总 runs 数，一个反应一场比赛中最大的 runs 数。下面的代码用 sum 语句实现总 run 数，用 retain 和 max 函数实现最大 runs 数：

```
* Using RETAIN and sum statements to find most runs and total runs;
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;
PROC PRINT DATA = gamestats;
   TITLE "Season's Record to Date";
RUN;
```

变量 maxruns 取前面迭代的 maxruns 和 runs 中最大值；变量 runstodate 将每一场比赛的 runs 都加到自己身上。结果如下：

Season's Record to Date 1

Obs	Month	Day	Team	Hits	Runs	Max Runs	Runs ToDate
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

3.10 用数组简化程序

对于太多变量要处理的程序，数组将大大简化程序。

SAS 中，数组是一组变量，变量可以是已存在的，也可以是新创建的。

数组在数据步中用 ARRAY 来定义，基本形式为：

ARRAY name(n) $ variable-list;

Name 是数组名，n 是变量数，()也可以用[]和{}代替。如果变量是字符串，则需要$，且变量是新创建的字符串时，$是必须的。变量名依照顺序排列，如数组：

ARRAY store(4) Macys Penneys Sears Target;

则 store(1)是 Macys，store(2)是 Penneys，store(3)是 Sear，store(4)是 Target。

数组本身不储存在数据集中，只有在数据步中才被定义。命名规则与变量一样（不超过 32 字节，以字母、下划线开头，只能包含字母、数字、下划线）

例子 广播电台 wbrk 做了一份关于歌曲的听众调查，对 10 首歌进行打分，分值在 1-5，如果没听过则填 9。数据文件 wbrk.dat 包括了被访者姓名、年龄、以及十首歌的打分。

```
Albany          54 4 3 5 9 9 2 1 4 4 9
Richmond        33 5 2 4 3 9 2 9 3 3 3
Oakland         27 1 3 2 9 9 9 3 4 2 3
Richmond        41 4 3 5 5 5 2 9 4 5 5
Berkeley        18 3 4 9 1 4 9 3 9 3 2
```

下面的代码将所有打分为 9 的改为缺失值：

```
* Change all 9s to missing values;
DATA songs;
   INFILE 'c:\MyRawData\WBRK.dat';
   INPUT City $ 1-15 Age domk wj hwow simbh kt aomm libm tr filp ttr;
   ARRAY song (10) domk wj hwow simbh kt aomm libm tr filp ttr;
   DO i = 1 TO 10;
      IF song(i) = 9 THEN song(i) = .;
   END;
PROC PRINT DATA = songs;
   TITLE 'WBRK Song Survey';
RUN;
```

十首歌被放入 song 的数组中。输出结果如下：

WBRK Song Survey 1

Obs	City	Age	domk	wj	hwow	simbh	kt	aomm	libm	tr	filp	ttr	i
1	Albany	54	4	3	5	.	.	2	1	4	4	.	11
2	Richmond	33	5	2	4	3	.	2	.	3	3	3	11
3	Oakland	27	1	3	2	.	.	.	3	4	2	3	11
4	Richmond	41	4	3	5	5	5	2	.	4	5	5	11
5	Berkeley	18	3	4	.	1	4	.	3	.	3	2	11

注意这里数组没有被保存到数据集中，而 i 被保存了。

3.11 列出变量名的快捷方式

如果想把 100 个变量放入数组，并不需要一个一个变量名的输入，有快捷方式可以列出变量名。

Number range lists 开始于同一个单词，结尾于连续的数字的，可以使用 Number range list。比如：

Variable list	Abbreviated list
`INPUT Cat8 Cat9 Cat10 Cat11 Cat12;`	`INPUT Cat8 - Cat12;`

Name range lists 这种列表是依据变量在数据集中的排列顺序来的，比如，创建如下数据步：

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

则变量的排列顺序就为：Y A C H R

那么可以依照这个顺序用“put 第一个变量--最后一个变量”来简化：

Variable list	Abbreviated list
`PUT y a c h r b;`	`PUT y -- b;`

如果不能确定数据集中变量的顺序，可以用 proc contents 的 postion 选项来查看。下面的代码列出了永久数据集 distance 的变量顺序：

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

Special sas name list

例子 广播电台 WBRK 想要修改前面的代码（将 9 改为缺失值），使用 mean 函数计算平均分数：

```
DATA songs;
   INFILE 'c:\MyRawData\WBRK.dat';
   INPUT City $ 1-15 Age domk wj hwow simbh kt aomm libm tr filp ttr;
   ARRAY new (10) Song1 - Song10;
   ARRAY old (10) domk -- ttr;
   DO i = 1 TO 10;
      IF old(i) = 9 THEN new(i) = .;
         ELSE new(i) = old(i);
   END;
   AvgScore = MEAN(OF Song1 - Song10);
PROC PRINT DATA = songs;
   TITLE 'WBRK Song Survey';
RUN;
```

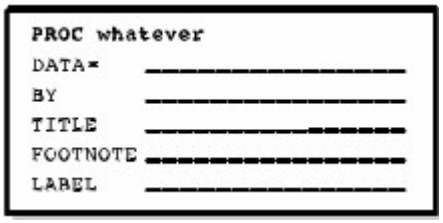
程序中，当原始变量（domk-ttr）值为 9 时，song 变量值为缺失值，否则就把原始变量的值赋给 song 变量。另外 avgscore 计算平均值：

WBRK Song Survey 1

Obs	City	Age	domk	wj	hwow	simbh	kt	aomm	libm	tr	filp	ttr	i	AvgScore
1	Albany	54	4	3	5	9	9	2	1	4	4	9	4	3.28571
2	Richmond	33	5	2	4	3	9	2	9	3	3	3	5	3.12500
3	Oakland	27	1	3	2	9	9	9	3	4	2	3	1	2.57143
4	Richmond	41	4	3	5	5	5	2	9	4	5	5	4	4.22222
5	Berkeley	18	3	4	9	1	4	9	3	9	3	2	3	2.85714

第四章 排序、打印并描述你的数据

4.1 使用 SAS 过程步



使用过程步就像填写一个如左图的表格，当然每个过程步都有独特的地方，本部分主要讨论各过程步相同的地方：

大部分过程步都有一个必须的语句，也有可选的语句，比如打印语句：proc print，这两个词是必须的，但可选的语句也有很多。

Proc 语句 所有的语句的必须部分为 proc+过程名，比如 print、contents 等。后面接一些可

选项。比如 proc print data=banana；
data=banana 选项告诉 SAS 打印哪个文件，如果不加，则 SAS 默认打印最近使用的数据。前面还可以家 libname 语句，建立一个对本地文件的链接（2.20），比如：

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

或者直接引用（2.21）：PROC CONTENTS DATA='c:\MySASLib\banana';

BY 语句 BY 语句只在过程 proc sort 中是必须的，它用来对观测值排序。其他过程 BY 告诉过程对变量进行分别分析，且是可选的。比如要对每个州进行分别分析，则为：BY State
另外，除了 proc sort，其他过程都假设了数据已经进行了排序，所以如果数据还没有排序，那么在分析之前要用 proc sort 排序。

TITLE 和 FOOTNOTE 语句 这是为输出加上标题和脚注。最基本的 title 语句为：title '标题'，双引号、单引号皆可，比如：

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

如果标题中带有撇号，则需用双引号，或者将撇号换为双撇号：

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

可以通过在 tile、footnote 后面加上数字来添加多个标题和脚注，

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

但是小数字的标题会代替大数字的标题，如 title2 会代替 title3。
标题的去处可以用 title+空值：TITLE;

Label 语句 它可以为输出的变量加上标签，一个标签最大 256 字节，下面的代码为 receivedate 和 shipdate 创建了标签：

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

注意的是，在数据步中使用 label 语句，则标签会保存在数据集中；在过程步中使用，标签只在这个过程中有效。

定制输出 使用系统选项，可以为输出设置诸如居中、日期、单行长度、页长度等。使用 Output Delivery System，还可以改变输出的风格，以不同的格式输出（HTML、RTF），甚至改变输出的任何细节。

输出数据集 可以用 ODS OUTPUT 语句为输出结果创立一个数据集（5.3），一些过程中也可以用 out=option。

4.2 用 where 语句在过程中构造子集

也可以用 where 构造子集，它方便快捷，因为他不创建新的数据集。且能够用在过程步中。
Where 语句的基本形式为：

```
WHERE condition;
```

只有满足条件的观测值才进行 proc 过程。
一些使用最多的操作符及例子：

Symbolic	Mnemonic	Example
=	EQ	WHERE Region = 'Spain';
¬=, ~=, ^=	NE	WHERE Region ~= 'Spain';
>	GT	WHERE Rainfall > 20;
<	LT	WHERE Rainfall < AvgRain;
>=	GE	WHERE Rainfall >= AvgRain + 5;
<=	LE	WHERE Rainfall <= 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');

例子 有一份关于画家的数据，artists.dat，包含画家的姓名、主要风格、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
```

第一步首先是数据步，读取数据、使用直接指代在 C 盘 mysaslib 目录下创建一个名为 style 的数据集。

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

某天如果想打印出印象派 impressionism 画家的情况，那么可以使用 where 语句

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

输出结果为：

```
                Major Impressionist Painters                1

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 用 proc sort 为数据排序

基本形式为：

```
PROC SORT;
BY variable-1...variable-n;
```

SAS 首先会按照第一个变量排序，再对后面的排序。

Data=，out=用来指定输入和输出数据，如果缺失 out=，则 SAS 会将排序后的数据集代替原来的数据集。下面的代码告诉 SAS 对数据 messy 排序，并将排序后的数据存在 neat 中：

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

选项 nodupkey 告诉 SAS 排序时删除重复值，比如：

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

SAS 默认是升序，可以用选项 DESCENDING 来变成降序，将 DESCENDING 加在要降序的变量前面：

```
BY State DESCENDING City;
```

例子 下面的数据显示了一些鲸鱼和鲨鱼品种的平均长度：

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

下面的代码读取并排序数据

```
DATA marine;
   INFILE 'c:\MyRawData\Sealife.dat';
   INPUT Name $ Family $ Length;
* Sort the data;
PROC SORT DATA = marine OUT = seasort NODUPKEY;
   BY Family DESCENDING Length;
PROC PRINT DATA = seasort;
   TITLE 'Whales and Sharks';
RUN;
```

输出结果为：

Whales and Sharks 1

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

因为 SAS 认为缺失值是比字符串和数值都小，所以排在了第一位。另外，由于 whale shark 40

的数据有两个，故因为 nodupkey 选项而被删除一个。说明可见日志：

```
NOTE: The data set WORK.MARINE has 11 observations and 3 variables.

NOTE: 1 observations with duplicate key values were deleted.
NOTE: The data set WORK.SEASORT has 10 observations and 3 variables.
```

4.4 用 proc print 打印你的数据

基本形式：PROC PRINT;

SAS 默认打印最近使用的数据集，DATA=可以指定数据集：

PROC PRINT DATA=data-set;

SAS 默认打印观测值数，noobs 选项可以取消。SAS 默认打印时用变量标签代替变量，用 label 可以改变取消：

PROC PRINT DATA=data-set NOOBS LABEL;

还有下面的选项：

BY variable-list;　前提是数据必须进行排序

ID variable-list;

SUM variable-list; 打印变量总数

VAR variable-list; 指定打印哪部分变量以及打印顺序，默认打印全部。

例子 有学生卖糖果的数据，Candy.dat，记录学生名、所属班级、销售日期、卖的糖果类型、卖出的糖果数。

```
Adriana     21   3/21/2000 MP  7
Nathan      14   3/21/2000 CD 19
Matthew     14   3/21/2000 CD 14
Claire      14   3/22/2000 CD 11
Caitlin     21   3/24/2000 CD  9
Ian         21   3/24/2000 MP 18
Chris       14   3/25/2000 CD  6
Anthony     21   3/25/2000 MP 13
Stephen     14   3/25/2000 CD 10
Erika       21   3/25/2000 MP 17
```

下面的程序读取数据、计算每个学生赚得的利润（每买一块赚 1.25 美元），并用 proc sort 按班级排序。接着在 proc print 语句中加入 by，以分班级打印，加入 sum，计算每个班级总利润：

```
DATA sales;
   INFILE 'c:\MyRawData\Candy.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;
```

输出结果为：

Candy Sales for Field Trip by Class 1

Class=14

Obs	Name	Date Returned	Candy Type	Profit
1	Nathan	14690	CD	23.75
2	Matthew	14690	CD	17.50
3	Claire	14691	CD	13.75
4	Chris	14694	CD	7.50
5	Stephen	14694	CD	12.50
Class				75.00

Class=21

Obs	Name	Date Returned	Candy Type	Profit
6	Adriana	14690	MP	8.75
7	Caitlin	14693	CD	11.25
8	Ian	14693	MP	22.50
9	Anthony	14694	MP	16.25
10	Erika	14694	MP	21.25
Class				80.00
				155.00

4.5 用 formats 改变打印外观

打印数据时，SAS 会自动为你安排最好的格式，小数点位数、空格等。
当不需要默认格式时，可以用 SAS formats 改变打印的外观。
对于字符串、数值、日期变量，SAS 有很多格式。比如可以用 commaw.d 格式打印有逗号的数字，用$w.格式控制打印的字符串数，用 MMDDYYw.格式将日期（以 1960.1.1 为基点的数字）打印成 12/03/2003 这样的格式。甚至可以将格式打印成十六进制、区位十进制、压缩十进制等。
SAS 格式的普通形式为：

Character	Numeric	Date
$formatw.	formatw.d	formatw.

符号说明：$说明了是字符串、format 是格式名、w 是包括包括在小数点在内的长度、d 是小数位数。句号非常重要，它用来区分格式名和变量名。
Format 语句 可以用 format 语句同时将格式和变量联系起来，用 format+变量名+格式名，比如想要将格式 DOLLAR8.2 和变量 profit、loss 联系起来，把格式 MMDDYY8.和格变量 saledate 联系起来：

```
FORMAT Profit Loss DOLLAR8.2 SaleDate MMDDYY8.;
```

Format 可以用在数据步和过程步中，前者将把格式永久储存，后者只是临时储存。
Put 语句 当写原始数据或者报告时，也可以在 put 语句中使用 formats，在每个变量后面加上格式：

```
PUT Profit DOLLAR8.2 Loss DOLLAR8.2 SaleDate MMDDYY8.;
```

例子 在上面的学生卖糖果的案例中，可以看到输出的日期是 SAS 日期值，这里用 format 变换成日期格式，并且用 DOLLAR6.2 将利润换成货币格式，

```
DATA sales;
   INFILE 'c:\MyRawData\Candy.dat';
   INPUT Name $ 1-11 Class @15 DateReturned MMDDYY10. CandyType $
         Quantity;
   Profit = Quantity * 1.25;
PROC PRINT DATA = sales;
   VAR Name DateReturned CandyType Profit;
   FORMAT DateReturned DATE9. Profit DOLLAR6.2;
   TITLE 'Candy Sale Data Using Formats';
RUN;
```

输出结果为：

Candy Sale Data Using Formats 1

Obs	Name	Date Returned	Candy Type	Profit
1	Adriana	21MAR2000	MP	$8.75
2	Nathan	21MAR2000	CD	$23.75
3	Matthew	21MAR2000	CD	$17.50
4	Claire	22MAR2000	CD	$13.75
5	Caitlin	24MAR2000	CD	$11.25
6	Ian	24MAR2000	MP	$22.50
7	Chris	25MAR2000	CD	$7.50
8	Anthony	25MAR2000	MP	$16.25
9	Stephen	25MAR2000	CD	$12.50
10	Erika	25MAR2000	MP	$21.25

4.6 可供选择的 formats

Format	Definition	Width range	Default width
Character			
$HEX*w*.	Converts character data to hexidecimal (specify *w* twice the length of the variable)	1-32767	4
$*w*.	Writes standard character data—does not trim leading blanks (same as $CHAR*w*.)	1-32767	Length of variable or 1
Date, Time, and Datetime[2]			
DATE*w*.	Writes SAS date values in form *ddmmmyy* or *ddmmmyyyy*	5-9	7
DATETIME*w.d*	Writes SAS datetime values in form *ddmmmyy:hh:mm:ss.ss*	7-40	16
DAY*w*.	Writes day of month from a SAS date value	2-32	2
EURDFDD*w*.	Writes a SAS date value in form: *dd.mm.yy*	2-10	8
JULIAN*w*.	Writes a Julian date from a SAS date value in form *yyddd* or *yyyyddd*	5-7	5
MMDDYY*w*.	Writes SAS date values in form *mmddyy* or *mmddyyyy*	2-10	8
TIME*w.d*	Writes SAS time values in form *hh:mm:ss.ss*	2-20	8
WEEKDATE*w*.	Writes SAS date values in form *day-of-week, month-name dd, yy* or *yyyy*	3-37	29
WORDDATE*w*.	Writes SAS date values in form *month-name dd, yyyy*	3-32	18
Numeric			
BEST*w*.	SAS chooses best format—this is the default format for writing numeric data	1-32	12
COMMA*w.d*	Writes numbers with commas separating every three digits	2-32	6
DOLLAR*w.d*	Writes numbers with a leading $ and commas separating every three digits	2-32	6
E*w*.	Writes numbers in scientific notation	7-32	12
PD*w.d*	Writes numbers in packed decimal—*w* specifies the number of bytes	1-16	1
w.d	Writes standard numeric data	1-32	none

下面是例子

Format	Input data	PUT statement	Results
Character			
$HEX*w*.	AB	PUT Name $HEX4.;	C1C2 (EBCDIC)[3] 4142 (ASCII)
$*w*.	my cat my snake	PUT Animal $8. '*';	my cat * my snak*
Date, Time, and Datetime			
DATE*w*.	8966	PUT Birth DATE7.; PUT Birth DATE9.;	19JUL84 19JUL1984
DATETIME*w*.	12182	PUT Start DATETIME13.; PUT Start DATETIME18.1;	01JAN60:03:23 01JAN60:03:23:02.0
DAY*w*.	8966	PUT Birth DAY2.; PUT Birth DAY7.;	19 19
EURDFDD*w*.	8966	PUT Birth EURDFDD8.;	19.07.84
JULIAN*w*.	8966	PUT Birth JULIAN5.; PUT Birth JULIAN7.;	84201 1984201
MMDDYY*w*.	8966	PUT Birth MMDDYY8.; PUT Birth MMDDYY6.;	7/19/84 071984
TIME*w.d*	12182	PUT Start TIME8.; PUT Start TIME11.2;	3:23:02 3:23:02.00
WEEKDATE*w*.	8966	PUT Birth WEEKDATE15.; PUT Birth WEEKDATE29.;	Thu, Jul 19, 84 Thursday, July 19, 1984
WORDDATE*w*.	8966	PUT Birth WORDDATE12.; PUT Birth WORDDATE18.;	Jul 19, 1984 July 19, 1984
Numeric			
BEST*w*.	1200001	PUT Value BEST6.; PUT Value BEST8.;	1.20E6 1200001
COMMA*w.d*	1200001	PUT Value COMMA9.; PUT Value COMMA12.2;	1,200,001 1,200,001.00
DOLLAR*w.d*	1200001	PUT Value DOLLAR10.; PUT Value DOLLAR13.2;	$1,200,001 $1,200,001.00
E*w*.	1200001	PUT Value E7.;	1.2E+06
PD*w.d*	128	PUT Value PD4.;	[4]
w.d	23.635	PUT Value 6.3; PUT Value 5.2;	23.635 23.64

4.7 使用 proc format 创建自己的格式

有时候变量值用数字代表实际的变量值，比如 1 代表男性，2 代表女性，这种代码在打印的时候不好解读，可以用 proc format 使得打印出想要的值。

基本形式为：

```
PROC FORMAT;
     VALUE name range-1 = 'formatted-text-1'
                range-2 = 'formatted-text-2'
                        .
                        .
                        .
                range-n = 'formatted-text-n';
```

Value 语句中的 name 是格式的名字，如果格式是位字符串设计，则必须以$开头，长度不能超过 32 个字节（包括$），不能以数字结尾，除了下划线，不能包含其他任何特殊符号。且名字不能与已有的格式名冲突。Range 是分配给等号右边文本的变量值，文本可以达到 32767 个字节，有的过程只会打印前面 8 或 16 个字节。下面是一个例子：

```
              'A' = 'Asia'
  1, 3, 5, 7, 9 = 'Odd'
  500000 - HIGH = 'Not Affordable'
       13 -< 20 = 'Teenager'
   0 <- HIGH = 'Positive Non Zero'
          OTHER = 'Bad Data'
```

变量值是字符串要加上引号，range 不止一个值要用逗号隔开，连续的 range 要用-，关键字 low 和 high 可以用来指代变量中最小和最大的的非缺失值。也可以用<来排除或指代某些范围，other 可以给任何没有列在 value 语句中的变量分配格式。

例子 有一份关于汽车公司客户的调查信息。包括客户年龄、性别（1 为男性，2 为女性）、每年收入、偏爱的汽车颜色（yellow,gray,blue,or white）：

```
19 1 14000 Y
45 1 65000 G
72 2 35000 B
31 1 44000 Y
58 2 83000 W
```

下面的代码读取数据，并使用 format 过程为颜色、性别和汽车创建格式，并在打印数据时用 format 为变量指定这些输出格式：

```
DATA carsurvey;
   INFILE 'c:\MyRawData\Cars.dat';
   INPUT Age Sex Income Color $;
PROC FORMAT;
   VALUE gender 1 = 'Male'
                2 = 'Female';
   VALUE agegroup 13 -< 20 = 'Teen'
                  20 -< 65 = 'Adult'
                  65 - HIGH = 'Senior';
   VALUE $col   'W' = 'Moon White'
                'B' = 'Sky Blue'
                'Y' = 'Sunburst Yellow'
                'G' = 'Rain Cloud Gray';
* Print data using user-defined and standard (DOLLAR8.) formats;
PROC PRINT DATA = carsurvey;
   FORMAT Sex gender. Age agegroup. Color $col. Income DOLLAR8.;
   TITLE 'Survey Results Printed with User-Defined Formats';
RUN;
```

输出结果为：

Survey Results Printed with User-Defined Formats 1

Obs	Age	Sex	Income	Color
1	Teen	Male	$14,000	Sunburst Yellow
2	Adult	Male	$65,000	Rain Cloud Gray
3	Senior	Female	$35,000	Sky Blue
4	Adult	Male	$44,000	Sunburst Yellow
5	Adult	Female	$83,000	Moon White

4.8 定制一个简单的报告

数据步可以帮助在报告中完成一些个性的需求，比如一页打印一个观测值等。
用 file 语句和 put 语句 ，基本形式为：

FILE‘file-specification’PRINT;

如 input，put 语句也有 list，column，formatted 方式，但因为 SAS 已经知道变量类型，因此不用符号$。且如果使用 list ，SAS 会自动在两个变量之间加上空格；使用 column 或者 formatted，SAS 将会把变量放在任何你指定的地方。使用指示器@n 指定移动到第 n 列，+n 指定移动 n 列，/跳动到下一行，#n 跳动到第 n 行。用@hold 住当前行。
例子 再一次使用学生卖糖果的案例，Candy.dat，记录学生名、所属班级、销售日期、卖的糖果类型、卖出的糖果数。

```
Adriana    21 3/21/2000 MP  7
Nathan     14 3/21/2000 CD 19
Matthew    14 3/21/2000 CD 14
Claire     14 3/22/2000 CD 11
Caitlin    21 3/24/2000 CD  9
Ian        21 3/24/2000 MP 18
Chris      14 3/25/2000 CD  6
Anthony    21 3/25/2000 MP 13
Stephen    14 3/25/2000 CD 10
Erika      21 3/25/2000 MP 17
```

老师想看每位学生的销售情况，故要每页分别打印一位学生的情况，代码如下：

```
* Write a report with FILE and PUT statements;
DATA _NULL_;
   INFILE 'c:\MyRawData\Candy.dat';
   INPUT Name $ 1-11 Class @15 DateReturned MMDDYY10.
         CandyType $ Quantity;
   Profit = Quantity * 1.25;
   FILE 'c:\MyRawData\Student.rep' PRINT;
   TITLE;

   PUT @5 'Candy sales report for ' Name 'from classroom ' Class
     // @5 'Congratulations!  You sold ' Quantity 'boxes of candy'
     / @5 'and earned ' Profit DOLLAR6.2 ' for our field trip.';
   PUT _PAGE_;
RUN;
```

Data null 是告诉 SAS 不要写数据集名，以便使得程序更快。File 语句创建了一个输出文件，空标题 title 语句告诉 SAS 去除所有的自动标题。
第一个 put 语句以一个指示器开头，@5，告诉 SAS 移动到第 5 列，接着打印出“candy sales report for”，后面是姓名 name。变量 name、class 和 quantity 都是以 list 方式打印，而 profit

是使用 formatted 方式打印，并给定格式 dollar6.2。一个斜杠是指跳到下一行，两个斜杠是跳到下两行。最后，语句 put_age_是在每个学生报告下面插上页码，程序运行后，日志说明如下：

```
NOTE: 10 records were read from the infile 'c:\MyRawData\Candy.dat'.

NOTE: 30 records were written to the file 'c:\MyRawData\Student.rep'.
```

前三页报告如下：

```
Candy sales report for Adriana from classroom 21

Congratulations!  You sold 7 boxes of candy
and earned  $8.75 for our field trip.
```

```
Candy sales report for Nathan from classroom 14

Congratulations!  You sold 19 boxes of candy
and earned $23.75 for our field trip.
```

```
Candy sales report for Matthew from classroom 14

Congratulations!  You sold 14 boxes of candy
and earned $17.50 for our field trip.
```

4.9 使用 proc means 描述数据

可以用 proc mens 查看一些简单的统计量，Means 过程开始于关键词 proc means，后面接需要打印的统计量，基本形式：

PROC MEANS options;

如果不加选项，则默认打印出非缺失值个数、均值、标准差、以及最大最小值，下面是用选项可以查看的统计量：

MAX	the maximum value
MIN	the minimum value
MEAN	the mean
MEDIAN	the median
N	number of non-missing values
NMISS	number of missing values
RANGE	the range
STDDEV	the standard deviation
SUM	the sum

如果没有其他语句，proc means 语句会给你数据集中所有观测值和所有数值变量的统计量，这里是一些可以用到的语句：

- BY variable-list; 分变量单独分析，但数据必须先按照 variable-list 的变量顺序排序（proc sort）。
- CLASS variable-list; 也是分变量单独分析，看起来会更集中一些，且不需要排序。
- VAR variable-list; 指定分析中使用哪种数值变量，默认则使用所有的数值变量。

例子 有一个花朵销售的数据，Flowers.dat，包括顾客 ID，销售日期，petunias，snapdragons，

marigolds 三种花的销售量：

```
756-01  05/04/2001 120  80 110
834-01  05/12/2001  90 160  60
901-02  05/18/2001  50 100  75
834-01  06/01/2001  80  60 100
756-01  06/11/2001 100 160  75
901-02  06/19/2001  60  60  60
756-01  06/25/2001  85 110 100
```

下面的代码读取数据，计算新变量销售月份，month，并使用 proc sort 按照月份排序，并使用 proc means 的 by 语句来按照月份描述数据：

```
DATA sales;
   INFILE 'c:\MyRawData\Flowers.dat';
   INPUT CustomerID $ @9 SaleDate MMDDYY10. Petunia SnapDragon
         Marigold;
   Month = MONTH(SaleDate);
PROC SORT DATA = sales;
   BY Month;
* Calculate means by Month for flower sales;
PROC MEANS DATA = sales;
   BY Month;
   VAR Petunia SnapDragon Marigold;
   TITLE 'Summary of Flower Sales by Month';
RUN;
```

输出结果为：

```
                    Summary of Flower Sales by Month                    1

---------------------------------- Month=5 ----------------------------------
                          The MEANS Procedure
Variable     N           Mean        Std Dev        Minimum        Maximum
————————————————————————————————————————————————————————————————————————
Petunia      3     86.6666667     35.1188458     50.0000000    120.0000000
SnapDragon   3    113.3333333     41.6333200     80.0000000    160.0000000
Marigold     3     81.6666667     25.6580072     60.0000000    110.0000000
————————————————————————————————————————————————————————————————————————

---------------------------------- Month=6 ----------------------------------
Variable     N           Mean        Std Dev        Minimum        Maximum
————————————————————————————————————————————————————————————————————————
Petunia      4     81.2500000     16.5201897     60.0000000    100.0000000
SnapDragon   4     97.5000000     47.8713554     60.0000000    160.0000000
Marigold     4     83.7500000     19.7378655     60.0000000    100.0000000
————————————————————————————————————————————————————————————————————————
```

4.10 将描述性统计写入 SAS 数据集中

有两种方法可以在 SAS 数据集中储存描述性统计量，Output Delivery System(ODS)，或者 output 语句。前者在 5.3，后者的基本形式为：

OUTPUT OUT=data-set output-statistic-list;

Data-set 是要储存结果的数据集名，output-statistic-list 则界定需要保存哪些统计量和名称，可能的形式为：

statistic(variable-list)=name-list

statistic 可能是 proc means 语句中的任何一种统计量（sum，n，mean…），variable-list 则界定 VAR 语句中哪些变量需要输出，name-list 则定义统计量的新名字。比如，proc means 语句产生了一个数据集 ZOOSUM，包括一个观测值和变量 lionweight（the mean of the lions'weights），BearWeight（the mean of the bears'weights）。

```
PROC MEANS DATA = zoo NOPRINT;
   VAR Lions Tigers Bears;
   OUTPUT OUT = zoosum MEAN(Lions Bears) = LionWeight BearWeight;
RUN;
```

Noprint 是告诉 SAS 不需要产生任何打印结果，因为已经将结果存入数据集中。

例子 仍然是花朵销售的数据

```
756-01  05/04/2001 120  80 110
834-01  05/12/2001  90 160  60
901-02  05/18/2001  50 100  75
834-01  06/01/2001  80  60 100
756-01  06/11/2001 100 160  75
901-02  06/19/2001  60  60  60
756-01  06/25/2001  85 110 100
```

要描述数据，每个顾客只有一个观测值，包括 SUM 和 MEAN，并且将结果储存到数据集中以便日后分析。下面的程序读取程序，按照 CustomerID 排序，使用 means 过程，结果存在 totals 数据集中。以原始名 Petunia,SnapDragon,and Marigold 给出 sum，以新变量名 MeanPetunia,MeanSnapDragon,and MeanMarigold 给出 mean

```
DATA sales;
   INFILE 'c:\MyRawData\Flowers.dat';
   INPUT CustomerID $ @9 SaleDate MMDDYY10. Petunia SnapDragon Marigold;
PROC SORT DATA = sales;
   BY CustomerID;
* Calculate means by CustomerID, output sum and mean to new data set;
PROC MEANS NOPRINT DATA = sales;
   BY CustomerID;
   VAR Petunia SnapDragon Marigold;
   OUTPUT OUT = totals  MEAN(Petunia SnapDragon Marigold) =
          MeanPetunia MeanSnapDragon MeanMarigold
      SUM(Petunia SnapDragon Marigold) = Petunia SnapDragon Marigold;
PROC PRINT DATA = totals;
   TITLE 'Sum of Flower Data over Customer ID';
   FORMAT MeanPetunia MeanSnapDragon MeanMarigold 3.;
RUN;
```

结果如下：

Sum of Flower Data over Customer ID 1

Obs	Customer ID	_TYPE_	_FREQ_	Mean Petunia	Mean Snap Dragon	Mean Marigold	Snap Petunia	Snap Dragon	Snap Marigold
1	756-01	0	3	102	117	95	305	350	285
2	834-01	0	2	85	110	80	170	220	160
3	901-02	0	2	55	80	68	110	160	135

4.11 用 proc freq 为数据计数

对一个变量计算频数叫做 one-way，两个叫做 two-way，多个叫做交叉表。使用 proc freq 最明显的目的是现实分类数据的分布情况，基本形式为：

```
PROC FREQ;
TABLES variable-combinations;
```

产生一维频率表，只要列出变量名。下面的语句列出了变量 yearseducation 的每一个值的个数。

```
TABLES YearsEducation;
```

建立两个变量的交叉表需要一个*号，下面的语句显示变量 Sex by YearsEducation 的频数情况：

```
TABLES Sex*YearsEducation;
```

这个语句之后可以用/option 的形式添加选项，主要下面几个：

LIST：用 list 形式打印交叉表（而不是网格）

MISSING：频率统计量中包含缺失值

NOCOL：强制在交叉表中不打印列百分比

NOROW：强制在交叉表中不打印行百分比

OUT=data-set：输出数据集

比如说，使用第二个选项：

```
TABLES Sex*YearsEducation/MISSING;
```

例子 有一家咖啡店的销售数据，记录了销售的咖啡种类（cappuccino,espresso,kona,or iced coffee），以及每次购买的顾客是打包还是原地就饮：

```
esp w cap d cap w kon w ice w kon d esp d kon w ice d esp d
cap w esp d cap d Kon d .   d kon w esp d cap w ice w kon w
kon w kon w ice d esp d kon w esp d esp w kon w cap w kon w
```

下面的代码就产生了一个 one-way 和 two-way 的频率表：

```
DATA orders;
   INFILE 'c:\MyRawData\Coffee.dat';
   INPUT Coffee $ Window $ @@;
* Print tables for Window and Window by Coffee;
PROC FREQ DATA = orders;
   TABLES Window  Window * Coffee;
   RUN;
```

代码告诉 SAS 打印两个表，一个是 one-way 的频率表，一个是交叉表。交叉表的每个小方格内，SAS 打印了频数、百分比、行百分比和列百分比。左边和右边是累积百分比。注意计算频数时没有考虑缺失值。

数据表的错误 kon？

The FREQ Procedure

Window	Frequency	Percent	Cumulative Frequency	Cumulative Percent
d	13	43.33	13	43.33
w	17	56.67	30	100.00

Table of Window by Coffee

Window / Coffee	Kon	cap	esp	ice	kon	Total
d	1	2	6	2	1	12
Percent	3.45	6.90	20.69	6.90	3.45	41.38
Row Pct	8.33	16.67	50.00	16.67	8.33	
Col Pct	100.00	33.33	75.00	50.00	10.00	
w	0	4	2	2	9	17
	0.00	13.79	6.90	6.90	31.03	58.62
	0.00	23.53	11.76	11.76	52.94	
	0.00	66.67	25.00	50.00	90.00	
Total	1	6	8	4	10	29
	3.45	20.69	27.59	13.79	34.48	100.00

Frequency Missing = 1

4.12 用 proc tabulate 产生一个表格报告

比起 print，means，freq，Proc tabulate 过程产生的报告更耐看。

Proc tabulate 的基本形式为：

```
PROC TABULATE;
CLASS classification-variable-list;
TABLE page-dimension,row-dimension,column-dimension;
```

Class 语句告诉 SAS 哪些变量将数据分成不同部分。

Table 语句可以定义一个表，可以用多个 table 语句定义多个表，

维度 table 语句可以在报告中指定三个维度：页、行、列。如果只指定一个维度，则默认是列维度；如果指定两个，则是行和列。

缺失数据 默认下不考虑缺失数据，在 proc 语句后面增加 missing 选项可以改变这种默认：

```
PROC TABULATE MISSING;
```

例子 有关于船的一些数据，Boats.dat，记录了每艘船的姓名、港口、移动方式（sailing 或者 power vesse）l，类型（schooner,catamaran,or yacht），使用它远行的价格

```
Silent Lady  Maalea  sail  sch 75.00
America II   Maalea  sail  yac 32.95
Aloha Anai   Lahaina sail  cat 62.00
Ocean Spirit Maalea  power cat 22.00
Anuenue      Maalea  sail  sch 47.50
Hana Lei     Maalea  power cat 28.99
Leilani      Maalea  power yac 19.99
Kalakaua     Maalea  power cat 29.50
Reef Runner  Lahaina power yac 29.95
Blue Dolphin Maalea  sail  cat 42.95
```

你想得到一份报告，包含了每一个港口的、sailing 或者 power vessel 的、每一种类型的、船

的数量，下面的代码用 proc tabulate 创建了一个三维报告：港口作为页、移动方式作为行、类型作为列：

```
DATA boats;
   INFILE 'c:\MyRawData\Boats.dat';
   INPUT Name $ 1-12 Port $ 14-20 Locomotion $ 22-26 Type $ 28-30
      Price 32-36;

* Tabulations with three dimensions;
PROC TABULATE DATA = boats;
   CLASS Port Locomotion Type;
   TABLE Port, Locomotion, Type;
   TITLE 'Number of Boats by Port, Locomotion, and Type';
RUN;
```

报告分两页，及港口的每个值情况为一页：

Number of Boats by Port, Locomotion, and Type 2

Port Maalea

	Type		
	cat	sch	yac
	N	N	N
Locomotion			
power	3.00	.	1.00
sail	1.00	2.00	1.00

4.13 为 proc tabulate 输出增加统计量

Class 语句列出分类变量，而 VAR 语句告诉 SAS 那些变量装的是连续数据。基本形式为：

```
PROC TABULATE;
VAR analysis-variable-list;
CLASS classification-variable-list;
TABLE page-dimension,row-dimension,column-dimension;
```

关键词 下面是 tabulate 可以计算的值：

ALL:增加行、列或页，显示总数

Max：最高值

Min：最低值

Mean：算术均值

Median：中位数

N：非缺失值个数

Nmiss：缺失值数

P90：90th 分位数

Pctn：某类的观测值百分数

Pctsum：某类值总和的百分数

STDDEV：标准差

SUM：求和

Concatenating,crossing,and grouping 维度、变量和关键词可以 Concatenating,crossing,and grouping，Concatenating 变量或关键词，只需用空格分开列出即可；cross 变量或关键字只需要用*分开列出即可；group 变量只需要用括号括住变量或关键词。

Concatenating:	TABLE Locomotion Type ALL;
Crossing:	TABLE MEAN*Price;
Crossing,grouping,and concatenating:	TABLE PCTN*(Locomotion Type);

例子 仍然是船的例子，

```
Silent Lady  Maalea  sail  sch 75.00
America II   Maalea  sail  yac 32.95
Aloha Anai   Lahaina sail  cat 62.00
Ocean Spirit Maalea  power cat 22.00
Anuenue      Maalea  sail  sch 47.50
Hana Lei     Maalea  power cat 28.99
Leilani      Maalea  power yac 19.99
Kalakaua     Maalea  power cat 29.50
Reef Runner  Lahaina power yac 29.95
Blue Dolphin Maalea  sail  cat 42.95
```

下面的代码类似 4.12，但多了 VAR 语句，table 只包括两维，但使用了 Concatenate,cross,and group：

```
DATA boats;
   INFILE 'c:\MyRawData\Boats.dat';
   INPUT Name $ 1-12 Port $ 14-20 Locomotion $ 22-26 Type $ 28-30
      Price 32-36;

* Tabulations with two dimensions and statistics;
PROC TABULATE DATA = boats;
   CLASS Locomotion Type;
   VAR Price;
   TABLE Locomotion ALL, MEAN*Price*(Type ALL);
   TITLE 'Mean Price by Locomotion and Type';
RUN;
```

输出结果如下：

Mean Price by Locomotion and Type 1

	Mean			
	Price			
	Type			
	cat	sch	yac	All
Locomotion				
power	26.83	.	24.97	26.09
sail	52.48	61.25	32.95	52.08
All	37.09	61.25	27.63	39.08

4.14 提升 proc tabulate 的输出外观

三种方式可以提升输出的外观：

Format=option 可以改变数据的格式，比如，在表中使得数字有逗号，并不含小数，则使用：

PROC TABULATE FORMAT=COMMA10.0;

Box=和 misstext=options format 只能用在 proc 语句中，而 box=和 misstext=只能用在 table 语句中。box=的作用是在 tabulete 报告的左上角的空格中写下一句简洁的语句（作用类似标题）。Misstext 则是位空数据格指定一个值，默认是一个句号，比如下句：

TABLE Region,MEAN*Sales/BOX='Mean Sales by Region' MISSTEXT='No Sales';

这是告诉 SAS 在左上角打印“Mean Sales by Region”，并且在没有数据的方格内打印“No Sales”

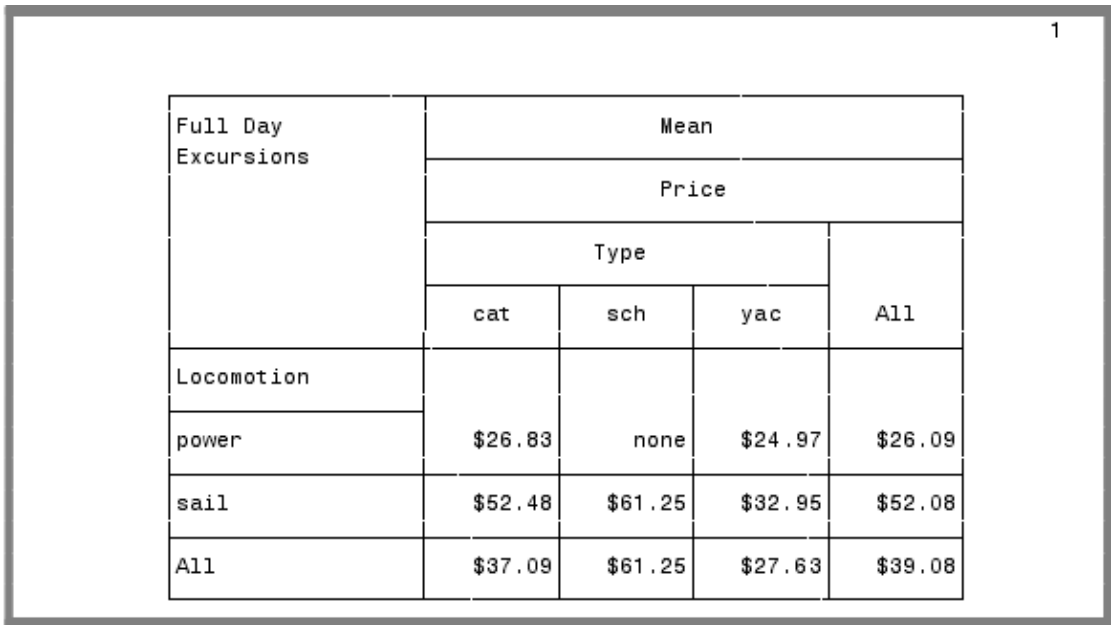
例子 仍然是船的数：

```
Silent Lady  Maalea  sail  sch 75.00
America II   Maalea  sail  yac 32.95
Aloha Anai   Lahaina sail  cat 62.00
Ocean Spirit Maalea  power cat 22.00
Anuenue      Maalea  sail  sch 47.50
Hana Lei     Maalea  power cat 28.99
Leilani      Maalea  power yac 19.99
Kalakaua     Maalea  power cat 29.50
Reef Runner  Lahaina power yac 29.95
Blue Dolphin Maalea  sail  cat 42.95
```

如下代码比前面多了 format、box、misstext 语句。注意 format 要出现在 proc 语句中，而 box 和 misstext 语句则出现在 table 语句中。

```
DATA boats;
   INFILE 'c:\MyRawData\Boats.dat';
   INPUT Name $ 1-12 Port $ 14-20 Locomotion $ 22-26 Type $ 28-30
      Price 32-36;

* PROC TABULATE report with options;
PROC TABULATE DATA = boats FORMAT=DOLLAR9.2;
   CLASS Locomotion Type;
   VAR Price;
   TABLE Locomotion ALL, MEAN*Price*(Type ALL)
      /BOX='Full Day Excursions' MISSTEXT='none';
   TITLE;
RUN;
```

这是“被提升了的”外观，由于 format 指定 dollar9.2，因此都用货币格式输出。左上角的 full day excursions 是由于 box 语句，空方格内的 none 是由于 misstext 语句。



1

Full Day Excursions	Mean			
	Price			
	Type			
	cat	sch	yac	All
Locomotion				
power	$26.83	none	$24.97	$26.09
sail	$52.48	$61.25	$32.95	$52.08
All	$37.09	$61.25	$27.63	$39.08

4.15 在 proc tabulate 输出的顶部

有两种方法可以改变顶部信息

Class 变量 变量值 要改变 class 语句列出的变量值的顶部，使用 format 创建一个用户定义的格式，然后用 format 语句将格式赋给变量。

变量名和关键字 改变变量名和关键字的顶部，用='text'赋值即可，可以用等号加空值的方法去除顶部，即=''，语句为：

TABLE Region='',MEAN=''*Sales='Mean Sales by Region';

这是告诉 SAS 移去 region 和 mean 的顶部，并且将 sale 的顶部换为“Mean Sales by Region”

有时候当行顶部被赋为空格时，会留下一个空白空格，可以用 row=float 强制去除这种空白空格： TABLE MEAN=''*Sales='Mean Sales by Region',Region=''/ROW=FLOAT;

例子 仍然是船的数据：

```
Silent Lady  Maalea  sail  sch 75.00
America II   Maalea  sail  yac 32.95
Aloha Anai   Lahaina sail  cat 62.00
Ocean Spirit Maalea  power cat 22.00
Anuenue      Maalea  sail  sch 47.50
Hana Lei     Maalea  power cat 28.99
Leilani      Maalea  power yac 19.99
Kalakaua     Maalea  power cat 29.50
Reef Runner  Lahaina power yac 29.95
Blue Dolphin Maalea  sail  cat 42.95
```

下面的代码和以前一样，多了对顶部的改变，format 语句创建了一个用户定义的格式$typ，并用 format 语句把这个格式赋给变量 type，table 语句中 locomotion、mean、type 的顶部被赋为空格，price 的顶部被赋值“Mean Price by Type of Boat.”

```
DATA boats;
   INFILE 'c:\MyRawData\Boats.dat';
   INPUT Name $ 1-12 Port $ 14-20 Locomotion $ 22-26 Type $ 28-30
      Price 32-36;

* Changing headers;
PROC FORMAT;
   VALUE $typ  'cat' = 'catamaran'
               'sch' = 'schooner'
               'yac' = 'yacht';

PROC TABULATE DATA = boats FORMAT=DOLLAR9.2;
   CLASS Locomotion Type;
   VAR Price;
   FORMAT Type $typ.;
   TABLE Locomotion='' ALL,
      MEAN=''*Price='Mean Price by Type of Boat'*(Type='' ALL)
      /BOX='Full Day Excursions' MISSTEXT='none';
   TITLE;
RUN;
```

输出结果为：

1

Full Day Excursions	Mean Price by Type of Boat			
	catamaran	schooner	yacht	All
power	$26.83	none	$24.97	$26.09
sail	$52.48	$61.25	$32.95	$52.08
All	$37.09	$61.25	$27.63	$39.08

这样的结果看起来清晰且紧凑。

4.16 为 proc tabulate 输出的数据方格指定多种格式

可以为不同变量指定不同格式，基本形式为：

variable-name*FORMAT=formatw.d

比如在 table 语句中插入这个复杂的语句：

TABLE Region,MEAN*(Sales*FORMAT=COMMA8.0 Profit*FORMAT=DOLLAR10.2);

这是给变量 sales 指定格式 comma8.0，给变量 profit 指定格式 dollar0.2

例子 仍然是船的数据，新增加了一个变量，以显示船的长度：

```
Silent Lady   Maalea   sail   sch 75.00   64
America II    Maalea   sail   yac 32.95   65
Aloha Anai    Lahaina sail   cat 62.00   60
Ocean Spirit  Maalea   power  cat 22.00   65
Anuenue       Maalea   sail   sch 47.50   52
Hana Lei      Maalea   power  cat 28.99  110
Leilani       Maalea   power  yac 19.99   45
Kalakaua      Maalea   power  cat 29.50   70
Reef Runner   Lahaina power  yac 29.95   50
Blue Dolphin  Maalea   sail   cat 42.95   65
```

假如你想在报告中同时 show 出平均价格和平均长度，仅为价格指定货币格式。下面的代码这样实现，为变量 price 指定格式 dollar6.2，为 length 指定格式 6.0：

```
DATA boats;
   INFILE 'c:\MyRawData\Boats.dat';
   INPUT Name $ 1-12 Port $ 14-20 Locomotion $ 22-26 Type $ 28-30
      Price 32-36 Length 38-40;

* Using the FORMAT= option in the TABLE statement;
PROC TABULATE DATA = boats;
   CLASS Locomotion Type;
   VAR Price Length;
   TABLE Locomotion ALL,
      MEAN * (Price*FORMAT=DOLLAR6.2 Length*FORMAT=6.0) * (Type ALL);
   TITLE 'Price and Length by Type of Boat';
RUN;
```

输出结果如下，注意价格和长度的格式不一样：

Price and Length by Type of Boat 1

	Mean							
	Price				Length			
	Type				Type			
	cat	sch	yac	All	cat	sch	yac	All
Locomotion								
power	$26.83	.	$24.97	$26.09	82	.	48	68
sail	$52.48	$61.25	$32.95	$52.08	63	58	65	61
All	$37.09	$61.25	$27.63	$39.08	74	58	53	65

4.17 用 proc report 产生一个简单的输出

Report 包含 print、means 和 tabulate、sort 的所有功能，可以用一本书来介绍，基本形式为：

PROC REPORT NOWINDOWS;

COLUMN variable-list;

Column 语句类似于 proc print 的 var 语句，告诉 SAS 哪些变量该包括并以何种顺序，如果遗漏语句 column，SAS 默认在数据集中包括所有变量，如果遗漏选项 nowINDOWS，SAS 默认启用交互 report 窗口。为使数据和顶部能很好的区分开来，可以使用 headline 和 headskip：

PROC REPORT NOWINDOWS HEADLINE HEADSKIP;

Headline 在顶部下面拉了一条线，headskip 在顶部下面留了一段空白。

数值变量 VS 字符串变量 从 proc report 得到的报告类型，部分依据于使用的数值类型。只要报告中起码有一个字符串变量，默认的报告就是每个观测值一行。但如果报告全是数值变量，默认 proc report 将会加总这些变量，即使是日期变量也会被加总。

例子 有一份关于美国国家公园（national parks）和国家纪念碑（monuments）的数据，Parks.dat，变量包括名字、类型（NP for national park or NM for national monument），地区（East or West），博物馆的数量，野营地的数量：

```
Dinosaur                NM West 2  6
Ellis Island            NM East 1  0
Everglades              NP East 5  2
Grand Canyon            NP West 5  3
Great Smoky Mountains   NP East 3  10
Hawaii Volcanoes        NP West 2  2
Lava Beds               NM West 1  1
Statue of Liberty       NM East 1  0
Theodore Roosevelt      NP .    2  2
Yellowstone             NP West 9  11
Yosemite                NP West 2  13
```

下面的代码形成了两份报告，第一份没有 column 语句，SAS 使用所有变量，第二份使用 column 语句，选择部分变量：

```
DATA natparks;
   INFILE 'c:\MyRawData\Parks.dat';
   INPUT Name $ 1-21 Type $ Region $ Museums Camping;

PROC REPORT DATA = natparks NOWINDOWS HEADLINE;
   TITLE 'Report with Character and Numeric Variables';
RUN;

PROC REPORT DATA = natparks NOWINDOWS HEADLINE;
   COLUMN Museums Camping;
   TITLE 'Report with Only Numeric Variables';
RUN;
```

第一份报告与 proc print 相似，第二份报告，由于只选择 museum 变量和 camping 两个数值型变量，默认直接显示加总情况：

Report with Character and Numeric Variables 1

Name	Type	Region	Museums	Camping
Dinosaur	NM	West	2	6
Ellis Island	NM	East	1	0
Everglades	NP	East	5	2
Grand Canyon	NP	West	5	3
Great Smoky Mountains	NP	East	3	10
Hawaii Volcanoes	NP	West	2	2
Lava Beds	NM	West	1	1
Statue of Liberty	NM	East	1	0
Theodore Roosevelt	NP		2	2
Yellowstone	NP	West	9	11
Yosemite	NP	West	2	13

Report with Only Numeric Variables 2

Museums	Camping
33	50

4.18 在 proc report 中使用 define 语句

Define 用来为单个变量指定一些选项，基本形式为：

DEFINE variable/options'column-header';

Usage 选项 这个选项告诉 SAS 如何使用这个变量，可能的 usage 选项包括：

Across：为变量的每一个变量值都创建一个列

Analysis：为变量创建统计量，数值变量默认有这个 usage 选项，且统计量默认为 sum。

Display：为数据集中的每一个观测值都创建一行，对于字符串变量，这个选项是默认的。

Group：为每个变量的变量值都创建一行。

Order：为每个观测值都创建一行，且行值的排列是是按照指定的变量来顺序。

改变列顶部 proc report 中几种方法可以改变列顶部，4.1 中的 label 语句，或者用 define 语句指定列顶部，下面的代码使得 SAS 的 report 按照 age 排序，并且以“Age at Admission”作为列顶部：

DEFINE Age / ORDER 'Age at/Admission';

缺失数据 默认在 order，group，和 across variables 中不考虑缺失值，用 missing 选项可以改变这种默认：

PROC REPORT NOWINDOWS MISSING;

例子 仍然是关于国家公园和纪念碑的数据，

```
Dinosaur                NM West 2  6
Ellis Island            NM East 1  0
Everglades              NP East 5  2
Grand Canyon            NP West 5  3
Great Smoky Mountains   NP East 3  10
Hawaii Volcanoes        NP West 2  2
Lava Beds               NM West 1  1
Statue of Liberty       NM East 1  0
Theodore Roosevelt      NP .    2  2
Yellowstone             NP West 9  11
Yosemite                NP West 2  13
```

下面的代码包含两个 define 语句，第一个用 order 选项来定义 region，第二个为变量 camping 定义列顶部。Camping 是一个数值变量，默认有 analysis 选项。Missing 选项也出现在了 proc 语句中，因此缺失值也会被考虑在报告中：

```
DATA natparks;
   INFILE 'c:\MyRawData\Parks.dat';
   INPUT Name $ 1-21 Type $ Region $ Museums Camping;

* PROC REPORT with ORDER variable, MISSING option, and column header;
PROC REPORT DATA = natparks NOWINDOWS HEADLINE MISSING;
   COLUMN Region Name Museums Camping;
   DEFINE Region / ORDER;
   DEFINE Camping / ANALYSIS 'Camp/Grounds';
   TITLE 'National Parks and Monuments Arranged by Region';
RUN;
```

输出结果为：

National Parks and Monuments Arranged by Region 1

Region	Name	Museums	Camp Grounds
	Theodore Roosevelt	2	2
East	Ellis Island	1	0
	Everglades	5	2
	Great Smoky Mountains	3	10
	Statue of Liberty	1	0
West	Dinosaur	2	6
	Grand Canyon	5	3
	Hawaii Volcanoes	2	2
	Lava Beds	1	1
	Yellowstone	9	11
	Yosemite	2	13

Region 有三个变量值，第一个是 missing 缺失值。

4.19 用 proc report 创建简易报告

Group 创建简易行，across 创建简易列。

Group 变量 下面的代码告诉 SAS 创建一个显示每个部门工资总和、奖金总和（数值变量将默认被加总）的报告：

Department	Salary	Bonus
A	~~~	~~
B	~~	~

```
PROC REPORT DATA = employees NOWINDOWS;
   COLUMN Department Salary Bonus;
   DEFINE Department / GROUP;
```

Across 变量 corss 变量，也需要 define 语句，不同的是，SAS 默认不是对变量值求和，而是计数。如果要加总，则需要再 across 变量和 analysis 变量之间加逗号，告诉 SAS 哪个变量要加总，下面的代码告诉 SAS 用列来显示出每个部门工资和奖金的总和：

Department			
A		B	
Salary	Bonus	Salary	Bonus
~~	~~	~~	~

```
PROC REPORT DATA = employees NOWINDOWS;
   COLUMN Department , (Salary Bonus);
   DEFINE Department / ACROSS;
```

例子 仍然是国家公园和纪念碑的例子，

```
Dinosaur                NM West 2  6
Ellis Island            NM East 1  0
Everglades              NP East 5  2
Grand Canyon            NP West 5  3
Great Smoky Mountains   NP East 3  10
Hawaii Volcanoes        NP West 2  2
Lava Beds               NM West 1  1
Statue of Liberty       NM East 1  0
Theodore Roosevelt      NP .    2  2
Yellowstone             NP West 9  11
Yosemite                NP West 2  13
```

下面的代码包含两个 proc report，第一个中，region 和 type 都被定义成 group 变量，第二个中，region 仍然是个 group 变量，但 type 是 across 变量。注意两个 column 语句基本一样，除了第二个中增加了标点（to cross the across variable with the analysis variables.）。

```
DATA natparks;
   INFILE 'c:\MyRawData\Parks.dat';
   INPUT Name $ 1-21 Type $ Region $ Museums Camping;

* Region and Type as GROUP variables;
PROC REPORT DATA = natparks NOWINDOWS HEADLINE;
   COLUMN Region Type Museums Camping;
   DEFINE Region / GROUP;
   DEFINE Type / GROUP;
   TITLE 'Summary Report with Two Group Variables';
RUN;

* Region as GROUP and Type as ACROSS with sums;
PROC REPORT DATA = natparks NOWINDOWS HEADLINE;
   COLUMN Region Type,(Museums Camping);
   DEFINE Region / GROUP;
   DEFINE Type / ACROSS;
   TITLE 'Summary Report with a Group and an Across Variable';
RUN;
```

输出结果为

Summary Report with Two Group Variables 1

Region	Type	Museums	Camping
East	NM	2	0
	NP	8	12
West	NM	3	7
	NP	18	29

Summary Report with a Group and an Across Variable 2

	Type			
	NM		NP	
Region	Museums	Camping	Museums	Camping
East	2	0	8	12
West	3	7	18	29

4.20 给 proc report 输出增加

Break 语句可以为报告增加停顿，为每个指定的变量的变量值增加停顿。基本形式如下：

BREAK location variable/options;

RBREAK location/options;

Location 有两种可能值——before 和 after，决定是之前停顿还是之后停顿。斜杠之后的选项告诉 SAS 插入哪种停顿，主要类型有：

OL 停顿的地方加入横线

Page 开始一个新的页面

Skip 插入一个空行

Summarize 插入数值变量之和

UL

需要注意的是，break 要求指定一个变量，而 rbreak 不需要。因为 rbreak 只产生一个停顿（开始或结尾），而 break 语句为指定的变量的每一个变量值都产生停顿。这个变量必须是 group 变量或 order 变量，并且要在 define 语句中定义过。可以在任何报告中使用 rbreak 语句，但只能在有最起码一个 group 或者 order 变量的报告中使用 break 语句。

例子 仍然是国家公园和纪念碑的例子：

```
Dinosaur                NM West 2  6
Ellis Island            NM East 1  0
Everglades              NP East 5  2
Grand Canyon            NP West 5  3
Great Smoky Mountains   NP East 3 10
Hawaii Volcanoes        NP West 2  2
Lava Beds               NM West 1  1
Statue of Liberty       NM East 1  0
Theodore Roosevelt      NP .    2  2
Yellowstone             NP West 9 11
Yosemite                NP West 2 13
```

下面的代码将 region 定义为 order 变量，使用 break 和 rbreak 语句和 after 选项，summarize

加总数值变量的和：

```
DATA natparks;
   INFILE 'c:\MyRawData\Parks.dat';
   INPUT Name $ 1-21 Type $ Region $ Museums Camping;

* PROC REPORT with breaks;
PROC REPORT DATA = natparks NOWINDOWS HEADLINE;
   COLUMN Name Region Museums Camping;
   DEFINE Region / ORDER;
   BREAK AFTER Region / SUMMARIZE OL SKIP;
   RBREAK AFTER / SUMMARIZE OL SKIP;
   TITLE 'National Parks';
RUN;
```

输出结果为：

National Parks 1

Name	Region	Museums	Camping
Ellis Island	East	1	0
Everglades		5	2
Great Smoky Mountains		3	10
Statue of Liberty		1	0
	East	10	12
Dinosaur	West	2	6
Grand Canyon		5	3
Hawaii Volcanoes		2	2
Lava Beds		1	1
Yellowstone		9	11
Yosemite		2	13
	West	21	36
		31	48

4.21 为 proc report 输出增加统计量

简单的方法是在 column 语句中加入统计量的关键字，常用的有：

Max、min、mean、median、n、nmiss、p90、pctn、pctsum、std、sum

给变量应用统计量 给变量应用统计量，在变量和统计量之间插入逗号即可，统计量 N 不需要逗号。如：

COLUMN Age,MEDIAN N;

为多个变量应用多个统计量，需要括号，如下面代码要求一个变量 age 应用两个统计量 min 和 max；两个变量 height 和 weight 应用一个统计量 mean：

COLUMN Age,(MIN MAX)(Height Weight),MEAN;

例子 仍然是国家公园和纪念碑的数据：

```
Dinosaur              NM West 2  6
Ellis Island          NM East 1  0
Everglades            NP East 5  2
Grand Canyon          NP West 5  3
Great Smoky Mountains NP East 3 10
Hawaii Volcanoes      NP West 2  2
Lava Beds             NM West 1  1
Statue of Liberty     NM East 1  0
Theodore Roosevelt    NP .    2  2
Yellowstone           NP West 9 11
Yosemite              NP West 2 13
```

下面的代码包括了两个 proc report，都应用了统计量 N 和 mean，但第一个定义 type 为 group 变量，第二个定义 type 为 across 变量。

```
DATA natparks;
   INFILE 'c:\MyRawData\Parks.dat';
   INPUT Name $ 1-21 Type $ Region $ Museums Camping;

*Statistics in COLUMN statement with two group variables;
PROC REPORT DATA = natparks NOWINDOWS HEADLINE;
   COLUMN Region Type N (Museums Camping),MEAN;
   DEFINE Region / GROUP;
   DEFINE Type / GROUP;
   TITLE 'Statistics with Two Group Variables';
RUN;

*Statistics in COLUMN statement with group and across variables;
PROC REPORT DATA = natparks NOWINDOWS HEADLINE;
   COLUMN Region N Type,(Museums Camping),MEAN;
   DEFINE Region / GROUP;
   DEFINE Type / ACROSS;
   TITLE 'Statistics with a Group and Across Variable';
RUN;
```

输出结果为：

```
              Statistics with Two Group Variables              1

                                        Museums    Camping
        Region    Type              N      MEAN       MEAN
        ------------------------------------------------------
        East      NM                2         1          0
                  NP                2         4          6
        West      NM                2       1.5        3.5
                  NP                4       4.5       7.25

           Statistics with a Group and Across Variable          2

                                           Type
                                NM                      NP
                          Museums    Camping    Museums    Camping
   Region            N       MEAN       MEAN       MEAN       MEAN
   -----------------------------------------------------------------
   East              4          1          0          4          6
   West              6        1.5        3.5        4.5       7.25
```

第五章 用 ODS 增强你的输出结果

5.1 ODC（Output Delivery System）的概念

过程步不会产生输出，它只会产生数据，然后把数据发送给 ODC，以决定输出的样式等。所以，不要考虑是否使用 ODC，而考虑怎么使用。

ODC 就像一家商务飞机，游客乘坐 car 和 bus 赶来，在机场确认行李、安检、最终登机，飞往目的地。Ods 中，数据就想游客，通过各种过程步而来，ODS 处理每一个数据集并发送到目的地。实际上，不同的 ODS 类型就是目的地，当达到目的地时，而数据的样式是由模板决定。

目的地 如果没有指定目的地，那么你的数据默认发往“列表 listing”，这里有几种可选的目的地

LISTING 标准 SAS 输出

Output SAS 输出数据集

Html 超文本标记语言

RTF 富文本格式

PRINTER high resolution printer output

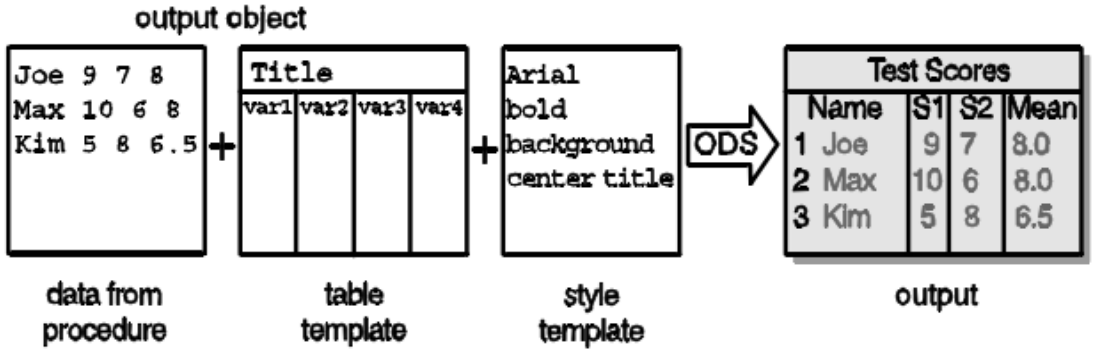
PS 附言

PCT Printer Control Language

PDF、MARKUP、DOCUMENT

DOCUMENT 目的地，允许创建一个可重复使用的输出。

风格和表模板 模板描述 ODS 如何制定数据格式并呈现数据。最普通的两个模板类型和是表模板类型和风格模板类型。表模板类型制定基本的输出结构，而风格模板类型制定输出将如何呈现。ODS 将过程产生的数据和和表模板结合成输出对象，输出对象接着与风格模板结合，并发送到目的地，创建出输出。



可以使用 template 过程创建自己的风格模板，但 proc template 过程晦涩难懂。幸运的是，有一个最简单和最快速的方法控制修改输出，即使用内置风格模板。可用 proc template 语句来访问内置模板：

```
PROC TEMPLATE;
LIST STYLES;
 RUN;
```

一些内置模板如下：

BARETTSBLUE	DEFAULT	PRINTER	SASWEB
BEIGE	D3D	RTF	SERIFPRINTER
BRICK	FANCYPRINTER	SANSPRINTER	STATDOC
BROWN	MINIMAL	SASDOCPRINTER	THEME

注意 RTF 和 PRINTER 既是目的地名又是风格名。DEFAULT 是 HTML 的默认风格，RTF 是 RTF 输出的默认风格，PRINTER 是 PRINTER 的默认风格。

Print、report、TABULATE 三个过程中，可以使用 style=option 来直接控制输出特征，而不需要创建一个新的模板。

5.2 追踪选择过程的输出

当 ODS 接受来自过程的的数据时，它将数据与表模板结合。对应的表模板和数据就叫做输出对象。如果使用 by 语句，SAS 会为每一个 BY 组产生一个输出对象。每一个输出对象都有名字，可以用 ODS TRACE 语句来查找，并用 ODS SELECT 语句来选择。

ODS TRACE 语句 ODS TRACE 语句告诉 SAS 打印出 SAS 日志中输出对象的信息。这里有两个 ODS TRACE 的语句，一个是打开 trace，一个是关闭。使用方法实例如下：

```
ODS TRACE ON;
the PROC steps you want to trace go here
RUN;
ODS TRACE OFF;
```

注意关闭语句要在 run 后面，否则在程序运行之前就关闭了 trace。

例子 有关于番茄种类的数据，包括每种番茄的名字、颜色、从播种到收获的天数、典型重量：

```
Big Zac, red, 80, 5
Delicious, red, 80, 3
Dinner Plate, red, 90, 2
Goliath, red, 85, 1.5
Mega Tom, red, 80, 2
Big Rainbow, yellow, 90, 1.5
Pineapple, yellow, 85, 2
```

下面代码创建了一个名为 giant 的数据集，并使用 ODS TRACE ON 和 ODS TRACE off 语句来追踪 proc means 过程。

```
DATA giant;
   INFILE 'c:\MyRawData\Tomatoes.dat' DSD;
   INPUT Name :$15. Color $ Days Weight;
* Trace PROC MEANS;
ODS TRACE ON;
PROC MEANS DATA = giant;
   BY Color;
RUN;
ODS TRACE OFF;
```

程序运行后，日志窗口中就会有如下的追踪（由于使用了 BY 语句，故按照 BY 的组来追踪）：

```
Output Added:
-------------
Name:        Summary
Label:       Summary statistics
Template:    base.summary
Path:        Means.ByGroup1.Summary
-------------
NOTE: The above message was for the following by-group: Color=red

Output Added:
-------------
Name:        Summary
Label:       Summary statistics
Template:    base.summary
Path:        Means.ByGroup2.Summary
-------------
NOTE: The above message was for the following by-group: Color=yellow
```

ODS select 语句 知道输出对象的名字之后，可以用 ODS SELECT 语句来选择需要的输出对象。基本形式为：

```
The PROC step with the output objects you want to select
ODS SELECT output-object-list;
RUN;
```

Output-object-list 是名字、标签、一个或更多的饿输出对象的路径。

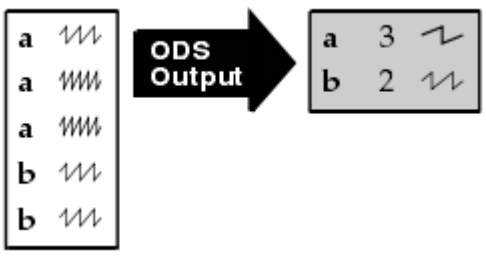
例子 下面代码对 giant 运行了 proc means，并用 ODS SELECT 语句选择了第一个输出对象，mean：

```
PROC MEANS DATA = giant;
   BY Color;
   TITLE 'Red Tomatoes';
ODS SELECT Means.ByGroup1.Summary;
RUN;
```

输出结果为：

```
                              Red Tomatoes                                1

--------------------------- Color=red ----------------------------

                        The MEANS Procedure

Variable  N          Mean       Std Dev       Minimum       Maximum
————————————————————————————————————————————————————————————————
Days      5    83.0000000     4.4721360    80.0000000    90.0000000
Weight    5     2.7000000     1.3964240     1.5000000     5.0000000
————————————————————————————————————————————————————————————————
```

5.3 从过程输出中创建 SAS 数据集



有时需要把一个过程的结果弄到 SAS 数据集中，有的过程用 output 或 out=实现。但用 ODS，可以储存从过程输出的任何一部分。首先要使用 ODS TRACE 语句决定选择输出对象名。然后使用 ODS OUTPUT 语句将输出对象发送到 OUTPUT 目

的地中。

ODS OUTPUT 语句 基本形式为：

ODS OUTPUT output-object=new-data-set;

这个语句不属于数据步和过程步。ODS OUTPUT 打开 SAS 数据集并等待正确的过程输出，数据集保持开放，直到过程步的结尾。因为 ODS OUTPUT 是立即执行的，它将应用于 proc 正在处理的数据，或者应用于下一个 proc（如果目前没有 proc）。为确保得到正确的输出，建议将 ODS OUTPUT 语句放在 PROC 语句之后，下一个 PROC 、DATA 或 RUN 语句之前。

例子 仍然是关于番茄的数据：

```
Big Zac, red, 80, 5
Delicious, red, 80, 3
Dinner Plate, red, 90, 2
Goliath, red, 85, 1.5
Mega Tom, red, 80, 2
Big Rainbow, yellow, 90, 1.5
Pineapple, yellow, 85, 2
```

下面是引用与 SAS 日志，显示由 proc tabulate 产生的追踪（trace），tabulate 产生一个叫做 table 的输出：

```
Output Added:
-------------
Name:        Table
Label:       Table 1
Data Name:   Report
Path:        Tabulate.Report.Table
--------------
```

下面的代码读取数据、使用 ODS OUTPUT 语句来创建叫做 TABOUT（来自 TABLE 输出对象）的 SAS 数据集，然后用 proc print 打印出新数据集。

```
DATA giant;
   INFILE 'c:\MyRawData\Tomatoes.dat' DSD;
   INPUT Name :$15. Color $ Days Weight;
PROC TABULATE DATA = giant;
   CLASS Color;
   VAR Days Weight;
   TABLE Color ALL, (Days Weight) * MEAN;
   TITLE 'Standard TABULATE Output';
ODS OUTPUT Table = tabout;
RUN;
PROC PRINT DATA = tabout;
   TITLE 'OUTPUT SAS Data Set from TABULATE';
RUN;
```

有两部分输出结果，第一部分是标准 tabular 结果，有 proc tabulate 产生。下面是 TABOUT 数据集，由 ODS OUTPUT 语句产生，并有 proc print 打印。

Standard TABULATE Output 1

	Days	Weight
	Mean	Mean
Color		
red	83.00	2.70
yellow	87.50	1.75
All	84.29	2.43

Output Data Set from TABULATE 2

Obs	Color	_TYPE_	_PAGE_	_TABLE_	Days_ Mean	Weight_ Mean
1	red	1	1	1	83.0000	2.70000
2	yellow	1	1	1	87.5000	1.75000
3		0	1	1	84.2857	2.42857

5.4 使用 ODS 语句创建 HTML 输出

将输出发送到 HTML 目的地，将得到 HTML 格式的文件。这个文件也可以被读入 spreadsheets，甚至被打印或导入到文字处理软件中（有些格式会发生变化）。总之，产生一个 HTML 文件只需两步语句——打开 HTML 文件、关闭。

ODS 语句 将输出发送到 HTML 目的地，使用 ODS HTML 语句，基本形式为：

```
ODS HTML BODY='body-filename.html' options;
```

Option 是用来改变 HTML 的类型(contents,page,or frame)，

- Contents= contents 文件是一个链接到主体文件的一个目录表，它将会列出输出的每个部分，点击表中某个条目，相关输出会出现。
- Page= page 文件类似于 contents 文件，不同的是，contents 通过标签列出输出的每个部分，而 page 文件通过页码列出。
- Frame= frame 允许同时访问在不同区域、框架或浏览器窗口中，访问主体文件、contents 文件和 page 文件。
- Style= 这个选项允许指定一个风格模板，默认的模板名为 default。

下面的语句告诉 SAS 发送一个输出给 HTML 目的地，储存一个名为 AnnualReport.html 的主体文件，并使用 D3D 风格。

```
ODS HTML BODY='AnnualReport.html' STYLE=D3D;
```

ODS 语句不属于数据步也不属于过程步，摆放它的好位置是 proc 过程步之前，这个过程步的输出正是你希望捕捉的。

关闭 HTML 文件的 ODS 语句为：

```
ODS HTML CLOSE;
```

将这个语句放在 proc 之后，接在 run 语句之后。

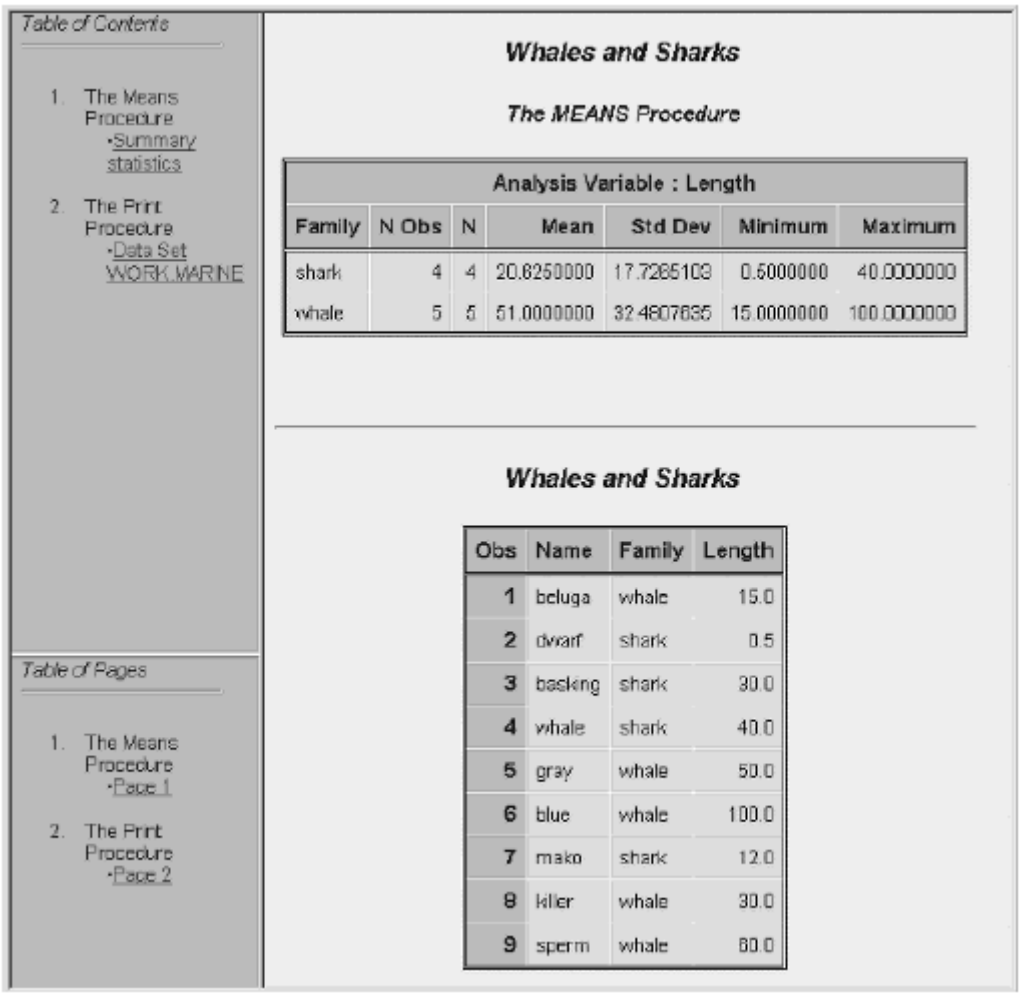
例子 下面的数据是关于选择的鲸鱼或鲨鱼的平均长度（英尺）

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

下面的代码创建了两个输出，一个是来自 means 过程，一个来自 print 过程。并且有两个 ODS 语句，一个创建了四个 HTML 文件：body,contents,page,and frame，一个是关闭 HTML 文件。

```
* Create the HTML files;
ODS HTML BODY = 'c:\MyHTMLFiles\MarineBody.html'
         CONTENTS = 'c:\MyHTMLFiles\MarineTOC.html'
         PAGE = 'c:\MyHTMLFiles\MarinePage.html'
         FRAME = 'c:\MyHTMLFiles\MarineFrame.html';
DATA marine;
   INFILE 'c:\MyRawData\Sealife.dat';
   INPUT Name $ Family $ Length @@;
PROC MEANS DATA = marine;
   CLASS Family;
   TITLE 'Whales and Sharks';
PROC PRINT DATA = marine;
RUN;
* Close the HTML files;
ODS HTML CLOSE;
```

输出结果如下：



5.5 使用 ODS 语句创建 RTF 输出

当你创建了富文本格式，可以将其发送到 word 中，并像 word 表格一样编辑修改。语句与 HTML 语句差不多，区别在于 option：

ODS 语句 ODS 打开 RTF 文件的基本形式为：

```
ODS RTF FILE='filename.rtf' options;
```

不像 HTML，RTF 文件只有一种类型，一些 option 如下：

- COLUMNS=n　要求一个柱状输出，n 是第几列。
- Bodytitle
- Sasdate　　这个选项告诉 SAS 当前的 SAS 会话开始运行时，使用日期和时间。
- Style=　　指定一个风格模板

下面的代码告诉 SAS 将输出发送到 RTF 目的地，储存一个名为 AnnualReport.rtf 的主体文件，并使用 FANCYPRINTER 风格。

```
ODS RTF FILE='AnnualReport.rtf' STYLE=FANCYPRINTER;
```

放置这个代码的较好位置也是在 proc 之前，而关闭语句也是放置在 proc 语句之后，接在 run 之后。

```
ODS RTF CLOSE;
```

例子 仍然是鲸鱼和鲨鱼平均重量的数据：

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

如下的代码从 means 和 print 中产生输出，程序中有两个 ODS 语句，一个打开 RTF 文件，一个关闭 RTF 文件。

```
* Create an RTF file;
ODS RTF FILE = 'c:\MyRTFFiles\Marine.rtf' BODYTITLE;
DATA marine;
   INFILE 'c:\MyRawData\Sealife.dat';
   INPUT Name $ Family $ Length @@;
PROC MEANS DATA = marine;
   CLASS Family;
   TITLE 'Whales and Sharks';
PROC PRINT DATA = marine;
RUN;
* Close the RTF file;
ODS RTF CLOSE;
```

Marine.RTF 文件在 word 中的结果如下，每一部分的输出结果在不同页面中出现：

The MEANS Procedure

Analysis Variable : Length						
Family	N Obs	N	Mean	Std Dev	Minimum	Maximum
shark	4	4	20.6250000	17.7265103	0.5000000	40.0000000
whale	6	6	50.8333333	29.0545464	15.0000000	100.0000000

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

5.6 使用 ODS 创建 printer 输出

ODS 语句 打开 printer 目的地的 ODS 语句最基本形式为：

ODS PRINTER;

如果使用这个简单的语句，SAS 将创建你先在系统需要的输出类型，并自动打印输出，而不是储存文件。可以用 add=option 来储存输出。类似 RTF，只有一种 PRINTER 类型文件。创建指定的输出类型的基本形式如下面几种：

- Default printer: ODS PRINTER FILE='filename.extension'options;
- PCL: ODS PCL FILE='filename.pcl'options;
- PDF: ODS PDF FILE='filename.pdf'options;
- PostScript: ODS PS FILE='filename.ps'options;

目的地一些有效的选项如下

- COLUMNS=n 要求一个柱状输出，n 是第几列。
- STYLE= 指定一个风格模板

下面的代码告诉 SAS 创建 PostScript output，将结果存在 AnnualReport.ps 中，并使用 FANCYPRINTER 风格。

ODS PS FILE='AnnualReport.ps' STYLE=FANCYPRINTER;

放置它的位置也是在 proc 之前。关闭一个 printer 文件 ODS 语句基本形式为;

ODS destination-name CLOSE;

destination-name 可能是 PRINTER,PCL,PDF,或者 PS，视开启语句中的目的地而定。放置在 proc 的 run 语句之后。

例子 仍然是鲸鱼和鲨鱼平均重量的数据：

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

如下的代码从 means 和 print 中产生输出，程序中有两个 ODS 语句，一个打开 PDF 文件，一个关闭 PDF 文件。

```
* Create the PDF file;
ODS PDF FILE = 'c:\MyPDFFiles\Marine.pdf';
DATA marine;
   INFILE 'c:\MyRawData\Sealife.dat';
   INPUT Name $ Family $ Length @@;
PROC MEANS DATA = marine;
   CLASS Family;
   TITLE 'Whales and Sharks';
PROC PRINT DATA = marine;
RUN;
* Close the PDF file;
ODS PDF CLOSE;
```

在 Adobe Acrobat 中的输出结果如下，每一部分的输出结果在不同页面中出现：

Whales and Sharks

The MEANS Procedure

Analysis Variable : Length						
Family	N Obs	N	Mean	Std Dev	Minimum	Maximum
shark	4	4	20.6250000	17.7265103	0.5000000	40.0000000
whale	6	6	50.8333333	29.0545464	15.0000000	100.0000000

Whales and Sharks

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

5.7 定制标题和注脚

通过在在 title 和 footnote 语句中插入一个简单的选项，可以改变注脚和标题的样式，title 和 footnote 语句的基本形式为:

TITLE options 'text-string-1' options 'text-string-2'…options 'text-string-n';

FOOTNOTE options 'text-string-1' options 'text-string-2 '…options'text-string-n';

可以将一段文字分成不同的部分，每个部分应用不同的样式，可以选择的主要选项如下表：

- COLOR= 为文本指定一种颜色
- BCOLOR= 为背景指定一种颜色
- HEIGHT= 为文本指定高度
- JUSTIFY= 要求对齐
- Font= 为文本指定字体
- Bold 粗体
- ITALIC 斜体

颜色 为一段文本不同部分指定不同的颜色

TITLE COLOR=BLACK 'Black' COLOR=GRAY 'Gray' COLOR=LTGRAY 'Light Gray';

显示为：

SAS 支持几百种颜色，但有的不能被 web 等识别，所以安全的颜色为：BLACK,BLUE,BROWN,CHARCOAL,CREAM,CYAN,GOLD,GRAY,GREEN,LILAC,LIME,MAGENTA,MAROON,OLIVE,ORANGE,PINK,PURPLE,RED,ROSE,SALMON,STEEL,TAN,VIOLET,WHITE,and YELLOW.

背景颜色 下面是用十六进制码来指定背景颜色：

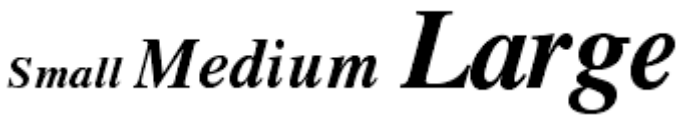
```
TITLE BCOLOR='#C0C0C0' 'This Title Has a Gray Background';
```

显示为：

高度 下面设置高度：

```
TITLE HEIGHT=12pt 'Small' HEIGHT=.25in 'Medium' HEIGHT=1cm 'Large';
```

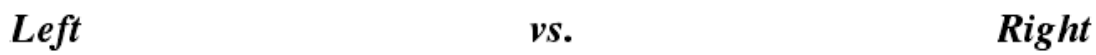
单位分别为像素、英尺、厘米，显示为：



对齐 下面分别设置左对齐、居中、右对齐：

```
TITLE JUSTIFY=LEFT 'Left' JUSTIFY=CENTER 'vs.' JUSTIFY=RIGHT 'Right';
```

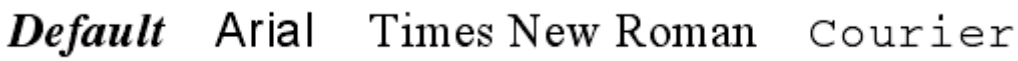
显示为：



字体 示例：

```
TITLE 'Default' FONT=Arial'Arial'
FONT='Times New Roman' 'Times New Roman' FONT=Courier'Courier';
```

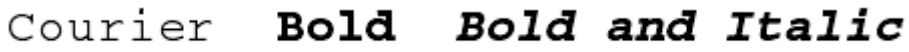
显示为：



加粗与斜体 默认下，标题是加粗且斜体的。关闭粗体和斜体要用 FONT=option，示例：

```
TITLE FONT=Courier'Courier'
BOLD'Bold'BOLD ITALIC'Bold and Italic';
```

显示为：



5.8 用 style=option 定制 proc print 输出

用 ODS 中的 style=option 可以改变输出的整体外观，但是如果要只改变头部，或者其中一列，要在 print、report 和 tabulate 过程中用 style=option。

Proc print 语句中使用 Style=option 的基本形式为：

```
PROC PRINT STYLE(location-list)={style-attribute=value};
```

location-list 说明了表中的哪一部分将应用风格，style-attribute 是要改变的风格属性，value 是属性值。下面的代码将 data 部分的 background 属性设为 pink：

```
PROC PRINT STYLE(DATA)={BACKGROUND=pink};
```

下面是可以指定改变风格的部分：

DATA	所有的数据单元
HEADER	列标题（变量名）
OBS	OBS 列，或者 ID 列（如果使用 ID 语句）的数据
OBSHEADER	OBS 列或 ID 列的标题
TOTAL	有 sum 语句产生的总和行的数据
GRANDTOTAL	

在 print 中放置 style=改变的是全表。比如 header 会改变全表的头部，如果只想改变某列的头部，需要再 VAR 语句中使用 style：

VAR variable-list/STYLE(location-list)={style-attribute=value};

仅有 variable-list 中的变量会被改变风格。想要不同的变量有不同的风格，可以使用复合 VAR 语句。

例子 下面是关于女子 5000 米滑冰奥运会金牌获得者的数据，变量一次为：奥运年年份、国家、时间、比赛记录（有 WR 的表示是世界记录）。

```
1988,Yvonne van Gennip,NED,7:14.13,WR
1992,Gunda Niemann,GER,7:31.57
1994,Claudia Pechstein,GER,7:14.37
1998,Claudia Pechstein,GER,6:59.61,WR
2002,Claudia Pechstein,GER,6:46.91,WR
```

下面的代码用 proc print 创建了 HTML 文件，使用的是默认风格模板。

```
ODS HTML FILE='c:\MyHTML\results.htm';
DATA skating;
  INFILE 'c:\MyData\women.csv' DSD MISSOVER;
  INPUT Year Name :$20. Country $
        Time $ Record $;
PROC PRINT DATA=skating;
  TITLE 'Women''s 5000 Meter Speed Skating';
  ID Year;
RUN;
ODS HTML CLOSE;
```

结果如下：

Women's 5000 Meter Speed Skating

Year	Name	Country	Time	Record
1988	Yvonne van Gennip	NED	7:14.13	WR
1992	Gunda Niemann	GER	7:31.57	
1994	Claudia Pechstein	GER	7:14.37	
1998	Claudia Pechstein	GER	6:59.61	WR
2002	Claudia Pechstein	GER	6:46.91	WR

下面的代码使用了 style 来改变所有数据单元的背景：

```
ODS HTML FILE='c:\MyHTML\results2.htm';
PROC PRINT DATA=skating
     STYLE(DATA)={BACKGROUND=white};
  TITLE 'Women''s 5000 Meter Speed Skating';
  ID Year;
RUN;
ODS HTML CLOSE;
```

结果如下：

Women's 5000 Meter Speed Skating

Year	Name	Country	Time	Record
1988	Yvonne van Gennip	NED	7:14.13	WR
1992	Gunda Niemann	GER	7:31.57	
1994	Claudia Pechstein	GER	7:14.37	
1998	Claudia Pechstein	GER	6:59.61	WR
2002	Claudia Pechstein	GER	6:46.91	WR

下面的代码增加 VAR 语句，将 record 列的字体改为斜体和粗体：

```
ODS HTML FILE='c:\MyHTML\results3.htm';
PROC PRINT DATA=skating
     STYLE(DATA)={BACKGROUND=white};
  TITLE 'Women''s 5000 Meter Speed Skating';
  VAR Name Country Time;
  VAR Record/STYLE(DATA)=
       {FONT_STYLE=italic FONT_WEIGHT=bold};
  ID Year;
RUN;
ODS HTML CLOSE;
```

结果为

Women's 5000 Meter Speed Skating

Year	Name	Country	Time	Record
1988	Yvonne van Gennip	NED	7:14.13	*WR*
1992	Gunda Niemann	GER	7:31.57	
1994	Claudia Pechstein	GER	7:14.37	
1998	Claudia Pechstein	GER	6:59.61	*WR*
2002	Claudia Pechstein	GER	6:46.91	*WR*

5.9 用 style=option 定制 proc report 输出

与 5.8 类似，基本语句为：

PROC REPORT STYLE(location-list)={style-attribute=value};

比如，如果想创建一个名为 MYSALES 的报告，并将列标题设置为绿色：

PROC REPORT DATA=mysales STYLE(HEADER)={BACKGROUND=green};

如果只需要改变报告中的某一列属性，则需要 define 语句，下面的语句告诉 SAS 使用 month 作为组变量，将其数据和标题的背景改为蓝色：

DEFINE Month/GROUP STYLE(HEADER COLUMN)={BACKGROUND=blue};

还可以用 break 和 rbreak 语句为摘要（summary）指定一个风格。下面的语句告诉 SAS，对于 month 的每一个值，为摘要使用红色背景，为总体报告摘要使用橙色背景：

```
BREAK AFTER Month / SUMMARIZE STYLE(SUMMARY) = {BACKGROUND = red};
RBREAK AFTER / SUMMARIZE STYLE(SUMMARY) = {BACKGROUND = orange};
```

例子 下面是不同的女子 5000 米滑冰奥运会金牌获得者的数据，变量依次为：姓名、国家、年份、金牌数。每一行包括了两条记录：

```
Lydia Skoblikova, URS, 1960, 2, Lydia Skoblikova, URS, 1964, 4
Karin Enke, GDR, 1980, 1, Karin Enke, GDR, 1984, 2
Christa Rothenburger, GDR, 1984, 1, Christa Rothenburger, GDR, 1988, 1
Bonnie Blair, USA, 1988, 1, Bonnie Blair, USA, 1992, 2
Gunda Nieman, GDR, 1992, 2, Bonnie Blair, USA, 1994, 2
Claudia Pechstein, GER, 1994, 1, Gunda Nieman, GDR, 1998, 1
Claudia Pechstein, GER, 1998, 1, Catriona LeMay, CAN, 1998, 1
Claudia Pechstein, GER, 2002, 2, Catriona LeMay, CAN, 2002, 1
```

下面的代码使用 proc report 创建了一个 HTML 文件，使用默认模板：

```
DATA skating;
   INFILE 'c:\MyRawData\speed.dat' DSD;
   INPUT Name :$20. Country $
      Year NumGold @@;

ODS HTML FILE='c:\MyHTML\speed.htm';
PROC REPORT DATA = skating NOWINDOWS;
   COLUMN Name Country NumGold, SUM;
   DEFINE Name / GROUP;
   DEFINE Country / GROUP;
   TITLE 'Olympic Women''s '
      'Speed Skating';
RUN;
ODS HTML CLOSE;
```

结果为：

Olympic Women's Speed Skating

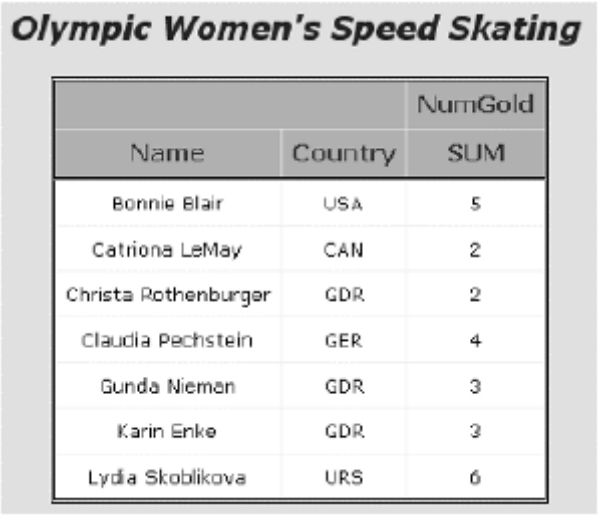
		NumGold
Name	Country	SUM
Bonnie Blair	USA	5
Catriona LeMay	CAN	2
Christa Rothenburger	GDR	2
Claudia Pechstein	GER	4
Gunda Nieman	GDR	3
Karin Enke	GDR	3
Lydia Skoblikova	URS	6

下面的代码使用 default 风格模板，但在 proc report 中增加 style 选项以改变所有数据的背景

颜色，并居中：

```
* STYLE= option in PROC statement;
ODS HTML FILE='c:\MyHTML\speed2.htm';
PROC REPORT DATA = skating NOWINDOWS
   STYLE(COLUMN) =
   {BACKGROUND = white JUST = center};
   COLUMN Name Country NumGold, SUM;
   DEFINE Name / GROUP;
   DEFINE Country / GROUP;
   TITLE 'Olympic Women''s '
      'Speed Skating';
RUN;
ODS HTML CLOSE;
```

结果为:



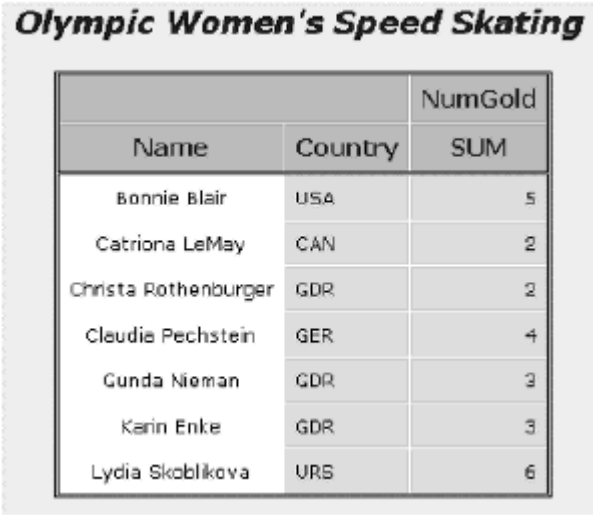
Olympic Women's Speed Skating

		NumGold
Name	Country	SUM
Bonnie Blair	USA	5
Catriona LeMay	CAN	2
Christa Rothenburger	GDR	2
Claudia Pechstein	GER	4
Gunda Nieman	GDR	3
Karin Enke	GDR	3
Lydia Skoblikova	URS	6

现在将 style 添加到 define 语句中，只改变 name 这一列：

```
* STYLE= option in DEFINE statement;
ODS HTML FILE='c:\MyHTML\speed3.htm';
PROC REPORT DATA = skating NOWINDOWS;
   COLUMN Name Country NumGold, SUM;
   DEFINE Name / GROUP
      STYLE(COLUMN) =
      {BACKGROUND = white JUST = center};
   DEFINE Country / GROUP;
   TITLE 'Olympic Women''s '
      'Speed Skating';
RUN;
ODS HTML CLOSE;
```

结果为：



Olympic Women's Speed Skating

		NumGold
Name	Country	SUM
Bonnie Blair	USA	5
Catriona LeMay	CAN	2
Christa Rothenburger	GDR	2
Claudia Pechstein	GER	4
Gunda Nieman	GDR	3
Karin Enke	GDR	3
Lydia Skoblikova	URS	6

5.10 使用 style=option 定制 proc tabulate 输出

下面显示了 proc tabulate 语句中可以应用 style 的地方，并且影响的表区域：

Statement	Table region affected
PROC TABULATE	all the data cells
CLASS	class variable name headings
CLASSLEV	class level value headings
TABLE (crossed with elements)[1]	element's data cell
VAR	analysis variable name headings

Proc tabulate 语句 如果想要来自数据集 MYSALES 的表的所有数据单元都变成黄色背景：

```
PROC TABULATE DATA=mysales STYLE={BACKGROUND=yellow};
```

Table 语句 下面的代码使得 all 列都有红色背景：

```
TABLE City,Month ALL*{STYLE={BACKGROUND=red}};
```

CLASSLEV,VAR,和 CLASS statements CLASSLEV,VAR,和 CLASS 语句都是在斜杠/后面应用 style 语句。注意的是，classlev 语句中的变量必须出现在 class 语句中，下面的代码是将月份 month 变量的每个值（Jan,Feb,Mar…）的前背景应用为绿色，使用 classlev 语句如下：

```
CLASSLEV Month/STYLE={FOREGROUND=green};
```

例子 下面是一份关于奥运会男子滑冰的数据，OR 是奥运会纪录，WR 是世界记录，NONE 既不是奥运会记录，也不是世界记录。注意一行有四个观测值：

```
1992 m500 None 1994 m500 OR 1998 m500 OR 2002 m500 OR
1992 m1000 None 1994 m1000 WR 1998 m1000 OR 2002 m1000 WR
1992 m1500 None 1994 m1500 WR 1998 m1500 WR 2002 m1500 WR
1992 m5000 None 1994 m5000 WR 1998 m5000 WR 2002 m5000 WR
1992 m10000 None 1994 m10000 WR 1998 m10000 WR 2002 m10000 WR
```

Tabulate 过程建立了一个表，以年份作为行，记录作为列。年份和 N 的顶部都通过设置为 null 将其消除。ODS 语句创建了 HTML 文件，使用默认模板：

```
ODS HTML FILE='c:\MyHTML\table.htm';
DATA skating;
  INFILE 'c:\MyData\records.dat';
  INPUT Year  Event $ Record $ @@;
PROC TABULATE DATA=skating;
  CLASS Year Record;
  TABLE Year='',Record*N='';
  TITLE 'Men''s Speed Skating';
  TITLE2 'Records Set at Olympics';
RUN;
ODS HTML CLOSE;
```

结果如下：

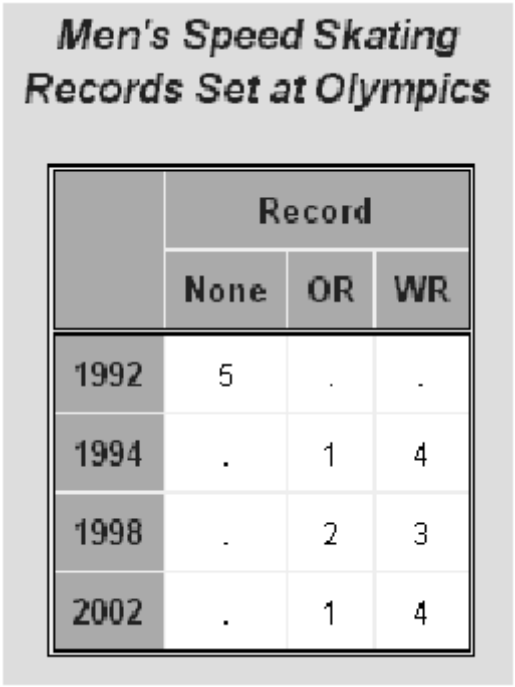
Men's Speed Skating
Records Set at Olympics

	Record		
	None	OR	WR
1992	5	.	.
1994	.	1	4
1998	.	2	3
2002	.	1	4

现在将数据单元的数据居中，并将背景设置为白色：

```
ODS HTML FILE='c:\MyHTML\table2.htm';
PROC TABULATE DATA=skating
     STYLE={JUST=center BACKGROUND=white};
  CLASS Year Record;
  TABLE Year='',Record*N='';
  TITLE 'Men''s Speed Skating';
  TITLE2 'Records Set at Olympics';
RUN;
ODS HTML CLOSE;
```

显示结果为：



Men's Speed Skating Records Set at Olympics

	Record		
	None	OR	WR
1992	5	.	.
1994	.	1	4
1998	.	2	3
2002	.	1	4

5.11 为你的输出增加交通信号灯

交通信号灯是基于单元格的值，来控制格的风格。它可以使得重要值变得醒目，它可以在 print、report、tabulate 中被使用。

使用之前需要做两件事：首先创建用户定义的格式。其次，在 style=中将风格属性等于你定义的格式，比如，你创建了一个格式：

```
PROC FORMAT;
  VALUE posneg
     LOW -< 0 = 'red'
     0-HIGH = 'black';
```

在 print 的 VAR 语句中，将属性值等于这个格式：

VAR Balance/STYLE={FOREGROUND=posneg.};

现在所有 balance 变量风格都发生变化。

例子 下面的数据是 2002 年冬奥会中，男子 5000 米滑冰前五名的数据，包括姓名、国家、成绩（用时）

```
1,Jochem Uytdehaage, Netherlands,374.66
2,Derek Parra, United States,377.98
3,Jens Boden, Germany,381.73
4,Dmitry Shepel, Russia,381.85
5,KC Boutiette, United States,382.97
```

下面代码读取打印数据，生成 HTML 文件，使用 default 模板：

```
ODS HTML FILE='c:\MyHTML\mens.html';
DATA results;
  INFILE
     'c:\MyRawData\mens5000.dat' DSD;
  INPUT Place Name :$20.
        Country :$15. Time ;
PROC PRINT DATA=results;
  ID Place;
  TITLE 'Men''s 5000m Speed Skating';
  TITLE2 '2002 Olympic Results';
RUN;
ODS HTML CLOSE;
```

结果为：

Men's 5000m Speed Skating
2002 Olympic Results

Place	Name	Country	Time
1	Jochem Uytdehaage	Netherlands	374.66
2	Derek Parra	United States	377.98
3	Jens Boden	Germany	381.73
4	Dmitry Shepel	Russia	381.85
5	KC Boutiette	United States	382.97

想要用信号灯显示每个成绩与世界记录的 378.72、奥运记录 382.20 比较的结果，先创建用户自定义的格式 REC，快于世界记录的用红色显示，橙色显示快于奥运记录的，其他颜色设置白色。接着在 print 语句中增加 var 语句，使用 style=option 为时间变量分配风格。最后，将定义的格式 REC 赋给 background。

```
ODS HTML FILE='c:\MyHTML\mens2.html';
PROC FORMAT;
  VALUE rec 0 -< 378.72 ='red'
            378.72 -< 382.20 = 'orange'
            382.20 - HIGH = 'white';
PROC PRINT DATA=results;
  ID Place;
  VAR Name Country;
  VAR Time/STYLE={BACKGROUND=rec.};
  TITLE 'Men''s 5000m Speed Skating';
  TITLE2 '2002 Olympic Results';
RUN;
ODS HTML CLOSE;
```

结果如下：

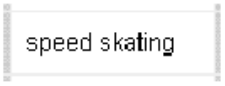
Men's 5000m Speed Skating
2002 Olympic Results

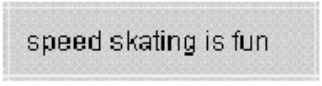
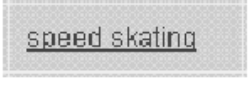
Place	Name	Country	Time
1	Jochem Uytdehaage	Netherlands	374.66
2	Derek Parra	United States	377.98
3	Jens Boden	Germany	381.73
4	Dmitry Shepel	Russia	381.85
5	KC Boutiette	United States	382.97

5.12 选择风格属性

Attribute	Description	Possible Values
BACKGROUND	Specifies the background color of the table or cell.	Any valid color[1]
BACKGROUNDIMAGE	Specifies a background image to be used for the table or cell. Not valid for RTF.	Any GIF, JPEG, or PNG image file[2]
FLYOVER	Specifies the pop-up text displayed when the cursor is held over the text (HTML) or if you double-click on the text (PDF).	Any text string enclosed in quotation marks
FONT_FACE	Specifies the font to use for the text in the cells.	Any valid font (Most devices support Times, Courier, Arial, and Helvetica)
FONT_SIZE	Specifies the relative size of the font for the text in cells.[3]	1 to 7
FONT_STYLE	Specifies the style of the font used in the cells.	ITALIC, ROMAN, or SLANT (Italic and slant may map to the same font)
FONT_WEIGHT	Specifies the weight of the font used in the cells.	BOLD, MEDIUM, or LIGHT
FOREGROUND	Specifies the color of the text in the cells.	Any valid color[1]

JUST	Specifies the justification of the text in the cells.	R|RIGHT, C|CENTER, L|LEFT, or D (decimal)
PRETEXT or POSTTEXT	Specifies text that goes either before (PRETEXT) or after (POSTTEXT) the text in the cells.	Any text string enclosed in quotation marks
PREIMAGE or POSTIMAGE	Specifies an image that will be inserted either before (PREIMAGE) or after (POSTIMAGE) the text in the cells.	Any GIF, JPEG or PNG image file (JPEG and PNG only for RTF)[2]
URL	Specifies the URL to link to from the text in the cell. HTML, PDF, and RTF only.	Any URL

Attribute	**STYLE= code**	**Result**
BACKGROUND	`STYLE(DATA)= {BACKGROUND=white};`	
BACKGROUNDIMAGE	`STYLE(DATA)= {BACKGROUNDIMAGE= 'c:\MyImages\snow.gif'};`	
FLYOVER	`STYLE(DATA)= {FLYOVER='Try it!'};`	
FONT_FACE	`STYLE(DATA)= {FONT_FACE=courier};`	
FONT_SIZE	`STYLE(DATA)= {FONT_SIZE=2};`	
FONT_STYLE	`STYLE(DATA)= {FONT_STYLE=italic};`	
FONT_WEIGHT	`STYLE(DATA)= {FONT_WEIGHT=bold};`	
FOREGROUND	`STYLE(DATA)= {FOREGROUND=white};`	

JUST	`STYLE(DATA)= {JUST=right};`	
PRETEXT or POSTTEXT	`STYLE(DATA)= {POST_TEXT=' is fun'};`	
PREIMAGE or POSTIMAGE	`STYLE(DATA)= {PREIMAGE='SS2.gif'};`	
URL	`STYLE(DATA)= {URL='http://skating.org'};`	

第六章 修改组合 SAS 数据集

6.1 使用 SET 语句修改数据集

SET 语句可以增加新变量、创建子集、修改数据集。SET 语句是一次一个变量地，将一个数据集放入数据步中予以分析。基本形式为：

```
DATA new-data-set;
SET data-set;
```

Data 语句指定了新数据集名，set 指定要读取的老数据集名。如果不想创建新的，则也可以在 data 中指定老数据集。

下面的代码创建了一个 Friday 的新数据集将 sales 数据集中的 day 属于 Friday 的观测值复制，并创建了新变量 total：

```
DATA friday;
   SET sales;
   IF Day = 'F';
   Total = Popcorn + Peanuts;
RUN;
```

例子 有一份关于火车运汽车的数据，汽车主为了在高峰期节省时间，或者为了节省汽油，选择让火车运汽车的方法，变量为一天中发车的时间、火车上的汽车数、火车中的人数：

```
10:10  6 21
12:15 10 56
15:30 10 25
11:30  8 34
13:15  8 12
10:45  6 13
20:30  6 32
23:15  6 12
```

数据被读入一个永久数据集 trains 中，储存在 MYSASLIB 目录文件夹下：

```
* Create permanent SAS data set trains;
DATA 'c:\MySASLib\trains';
   INFILE 'c:\MyRawData\Train.dat';
   INPUT Time TIME5. Cars People;
RUN;
```

由于每辆车的最大乘客数为 6 人，现在想知道一列火车上，平均每两汽车的乘客数是多少，可以在数据中插入一列，但这不在原始数据中计算，而是在一个新数据集中计算：

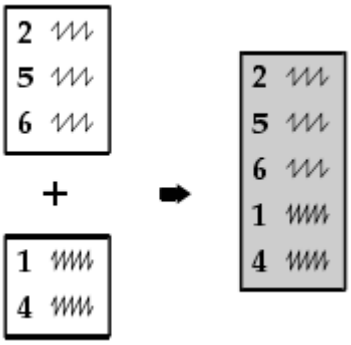
```
* Read the SAS data set trains with a SET statement;
DATA averagetrain;
   SET 'c:\MySASLib\trains';
   PeoplePerCar = People / Cars;
PROC PRINT DATA = averagetrain;
   TITLE 'Average Number of People per Train Car';
   FORMAT Time TIME5.;
RUN;
```

结果如下：

Average Number of People per Train Car 1

Obs	Time	Cars	People	People PerCar
1	10:10	6	21	3.50000
2	12:15	10	56	5.60000
3	15:30	10	25	2.50000
4	11:30	8	34	4.25000
5	13:15	8	12	1.50000
6	10:45	6	13	2.16667
7	20:30	6	32	5.33333
8	23:15	6	12	2.00000

6.2 使用 set 语句堆叠数据



运用 set 语句可以把一个数据集堆在另一个数据集上，适用于两个变量相同的两个数据集。

基本形式为：

```
DATA new-data-set;
SET data-set-1 data-set-n;
```

首先指定一个新的数据集，然后列出需要合并的旧数据集。如果一个数据集包含了另一个数据集没有的变量，那么合并后，该变量下将会出现缺失值。

例子 有如下两份南北数据，北方数据比南方多了一行变量（最后一行），其他变量均相同：

Data for South Entrance

```
S 43 3 27
S 44 3 24
S 45 3  2
```

Data for North Entrance

```
N 21 5 41 1
N 87 4 33 3
N 65 2 67 1
N 66 2  7 1
```

下面有三段代码，前两段将南方和北方的数据各输入数据集，并打印。第三段使用 SET 语句将南北方数据合并，并创建了新变量，AmountPaid：

```
DATA southentrance;
   INFILE 'c:\MyRawData\South.dat';
   INPUT Entrance $ PassNumber PartySize Age;
PROC PRINT DATA = southentrance;
   TITLE 'South Entrance Data';

DATA northentrance;
   INFILE 'c:\MyRawData\North.dat';
   INPUT Entrance $ PassNumber PartySize Age Lot;
PROC PRINT DATA = northentrance;
   TITLE 'North Entrance Data';
```

```
* Create a data set, both, combining northentrance and southentrance;
* Create a variable, AmountPaid, based on value of variable Age;
DATA both;
   SET southentrance northentrance;
   IF Age = . THEN AmountPaid = .;
      ELSE IF Age < 3  THEN AmountPaid = 0;
      ELSE IF Age < 65 THEN AmountPaid = 35;
      ELSE AmountPaid = 27;
PROC PRINT DATA = both;
   TITLE 'Both Entrances';
RUN;
```

输出结果如下：

South Entrance Data 1

Obs	Entrance	Pass Number	Party Size	Age
1	S	43	3	27
2	S	44	3	24
3	S	45	3	2

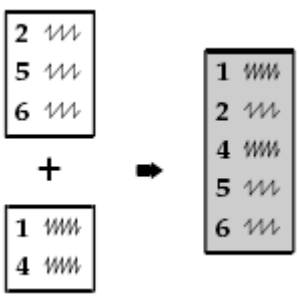
North Entrance Data 2

Obs	Entrance	Pass Number	Party Size	Age	Lot
1	N	21	5	41	1
2	N	87	4	33	3
3	N	65	2	67	1
4	N	66	2	7	1

Both Entrances 3

Obs	Entrance	Pass Number	Party Size	Age	Lot	Amount Paid
1	S	43	3	27	.	35
2	S	44	3	24	.	35
3	S	45	3	2	.	0
4	N	21	5	41	1	35
5	N	87	4	33	3	35
6	N	65	2	67	1	27
7	N	66	2	7	1	35

6.3 使用 SET 语句插入数据集



前面的堆叠数据，可能把数据顺序打乱，当然可以再用 proc sort 再将数据排序。但这可能效率低下。在 set 语句中使用 by 语句可以高效率的将数据按顺序合并，基本形式为;

```
DATA new-data-set;
    SET data-set-1 data-set-n;
  BY variable-list;
```

BY 语句中，可以指定一个或多个变量，让 SAS 进行排序。

注意，在合并几个数据之前，每个数据都要用 BY 进行排序，如果没有，则要用 proc sort 完成。

例子 仍然是刚才的例子：

Data for South Entrance

```
S 43 3 27
S 44 3 24
S 45 3  2
```

Data for North Entrance

```
N 21 5 41 1
N 87 4 33 3
N 65 2 67 1
N 66 2  7 1
```

注意南方的数据已经按照 pass number（第二个变量）进行了排序，北方的没有。下面还是三段代码，第一段对南方的数据输入，打印。第二段对北方的数据输入、排序、打印。第三段进行合并，并创建新变量 INTERLEAVE。

```
DATA southentrance;
   INFILE 'c:\MyRawData\South.dat';
   INPUT Entrance $ PassNumber PartySize Age;
PROC PRINT DATA = southentrance;
   TITLE 'South Entrance Data';

DATA northentrance;
   INFILE 'c:\MyRawData\North.dat';
   INPUT Entrance $ PassNumber PartySize Age Lot;
PROC SORT DATA = northentrance;
   BY PassNumber;
PROC PRINT DATA = northentrance;
   TITLE 'North Entrance Data';

* Interleave observations by PassNumber;
DATA interleave;
   SET northentrance southentrance;
   BY PassNumber;
PROC PRINT DATA = interleave;
   TITLE 'Both Entrances, By Pass Number';
RUN;
```

下面是输出结果：

South Entrance Data 1

Obs	Entrance	Pass Number	Party Size	Age
1	S	43	3	27
2	S	44	3	24
3	S	45	3	2

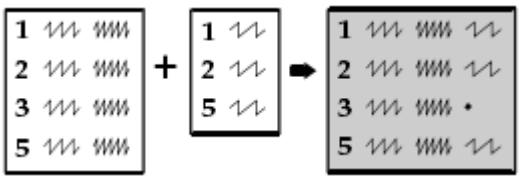
North Entrance Data 2

Obs	Entrance	Pass Number	Party Size	Age	Lot
1	N	21	5	41	1
2	N	65	2	67	1
3	N	66	2	7	1
4	N	87	4	33	3

Both Entrances, By Pass Number 3

Obs	Entrance	Pass Number	Party Size	Age	Lot
1	N	21	5	41	1
2	S	43	3	27	.
3	S	44	3	24	.
4	S	45	3	2	.
5	N	65	2	67	1
6	N	66	2	7	1
7	N	87	4	33	3

6.4 一对一匹配合并数据集



合并数据集，首先，如果数据没有排序，使用 sort 过程按照匹配变量排序。之后，在 data 语句中对新 SAS 数据集命名，再使用 merge 语句列出要合并的数据集名。使用 BY 语句说明共同变量。

```
DATA new-data-set;
    MERGE data-set-1 data-set-2;
    BY variable-list;
```

注意，如果两个数据集有重叠的变量——除了 BY 变量，那么第二个数据集中的此变量会覆盖第一个数据集中的。

例子 有一个巧克力店记录了每天所卖巧克力的类型以及数量，第一个文件 sales data 记录了所卖的巧克力代码和数量，第二个记录了巧克力代码、所代表的类型、描述。

Sales data

```
C865 15
K086  9
A536 21
S163 34
K014  1
A206 12
B713 29
```

Descriptions

```
A206 Mokka      Coffee buttercream in dark chocolate
A536 Walnoot    Walnut halves in bed of dark chocolate
B713 Frambozen  Raspberry marzipan covered in milk chocolate
C865 Vanille    Vanilla-flavored rolled in ground hazelnuts
K014 Kroon      Milk chocolate with a mint cream center
K086 Koning     Hazelnut paste in dark chocolate
M315 Pyramide   White with dark chocolate trimming
S163 Orbais     Chocolate cream in dark chocolate
```

下面有三段代码，前两段读取 sales 数据，description 数据。后者已经对 codenum 变量进行排序，前者需要用 proc sort 进行排序。否则会出现错误的信息：ERROR:BY variables are not properly sorted

```
DATA descriptions;
   INFILE 'c:\MyRawData\chocolate.dat' TRUNCOVER;
   INPUT CodeNum $ 1-4 Name $ 6-14 Description $ 15-60;
DATA sales;
   INFILE 'c:\MyRawData\chocsales.dat';
   INPUT CodeNum $ 1-4 PiecesSold 6-7;
PROC SORT DATA = sales;
   BY CodeNum;

* Merge data sets by CodeNum;
DATA chocolates;
   MERGE sales descriptions;
   BY CodeNum;
PROC PRINT DATA = chocolates;
   TITLE "Today's Chocolate Sales";
RUN;
```

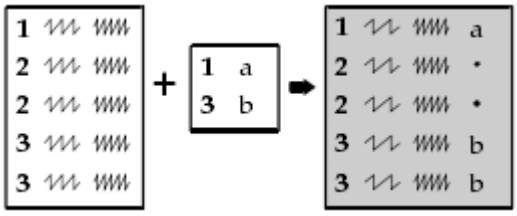
结果如下：

Today's Chocolate Sales 1

Obs	Code Num	Pieces Sold	Name	Description
1	A206	12	Mokka	Coffee buttercream in dark chocolate
2	A536	21	Walnoot	Walnut halves in bed of dark chocolate
3	B713	29	Frambozen	Raspberry marzipan covered in milk chocolate
4	C865	15	Vanille	Vanilla-flavored rolled in ground hazelnuts
5	K014	1	Kroon	Milk chocolate with a mint cream center
6	K086	9	Koning	Hazelnut paste in dark chocolate
7	M315	.	Pyramide	White with dark chocolate trimming
8	S163	34	Orbais	Chocolate cream in dark chocolate

注意 K086 的销售记录缺失，因为 sales data 中 没有关于其的记录。

6.5 一对多匹配合并数据



一对多合并是指一个数据集中的一个观测值可以与另一个数据集中的多个观测值匹配。基本形式与一对一一样：

```
DATA new-data-set;
   MERGE data-set-1 data-set-2;
   BY variable-list;
```

注意数据集的顺序，一对多的一要放在前面。在进行合并之前，仍然要对两个数据集按照匹配变量进行排序。其他注意与 6.4 差不多。

例子 有一份关于鞋子打折的数据，其中训练鞋、跑步鞋、走路鞋的折扣各不同。第一份数据是关于鞋子的风格、类型、价格。第二份数据是关于每个类型鞋子的折扣：

Shoes data

```
Max Flight       running 142.99
Zip Fit Leather  walking  83.99
Zoom Airborne    running 112.99
Light Step       walking  73.99
Max Step Woven   walking  75.99
Zip Sneak        c-train  92.99
```

Discount data

```
c-train .25
running .30
walking .20
```

下面的代码用多对一合并了两个数据：

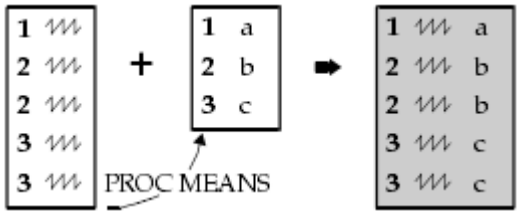
```
DATA regular;
   INFILE 'c:\MyRawData\Shoe.dat';
   INPUT Style $ 1-15 ExerciseType $ RegularPrice;
PROC SORT DATA = regular;
   BY ExerciseType;

DATA discount;
   INFILE 'c:\MyRawData\Disc.dat';
   INPUT ExerciseType $ Adjustment;

* Perform many-to-one match merge;
DATA prices;
   MERGE regular discount;
   BY ExerciseType;
   NewPrice = ROUND(RegularPrice - (RegularPrice * Adjustment), .01);
PROC PRINT DATA = prices;
   TITLE 'Price List for May';
RUN;
```

结果如下：

Price List for May 1

Obs	Style	Exercise Type	Regular Price	Adjustment	New Price
1	Zip Sneak	c-train	92.99	0.25	69.74
2	Max Flight	running	142.99	0.30	100.09
3	Zoom Airborne	running	112.99	0.30	79.09
4	Zip Fit Leather	walking	83.99	0.20	67.19
5	Light Step	walking	73.99	0.20	59.19
6	Max Step Woven	walking	75.99	0.20	60.79

6.6 合并统计量与原始数据



当你想比较每一个观测值和一组变量的均值时，可以先使用 proc means 计算统计量，并保存输出文件，再与原始文件合并。

例子 有一份关于鞋子销量的数据，变量为鞋子风格、类型、销量。现在想列出每种类型鞋子里，各风格的销售量所占的百分比：

```
Max Flight       running 1930
Zip Fit Leather  walking 2250
Zoom Airborne    running 4150
Light Step       walking 1130
Max Step Woven   walking 2230
Zip Sneak        c-train 1190
```

代码：

```
DATA shoes;
   INFILE 'c:\MyRawData\Shoesales.dat';
   INPUT Style $ 1-15 ExerciseType $ Sales;
PROC SORT DATA = shoes;
   BY ExerciseType;

* Summarize sales by ExerciseType and print;
PROC MEANS NOPRINT DATA = shoes;
   VAR Sales;
   BY ExerciseType;
   OUTPUT OUT = summarydata SUM(Sales) = Total;
PROC PRINT DATA = summarydata;
   TITLE 'Summary Data Set';

* Merge totals with the original data set;
DATA shoesummary;
   MERGE shoes summarydata;
   BY ExerciseType;
   Percent = Sales / Total * 100;
PROC PRINT DATA = shoesummary;
   BY ExerciseType;
   ID ExerciseType;
   VAR Style Sales Total Percent;
   TITLE 'Sales Share by Type of Exercise';
RUN;
```

描述统计量的结果 summarydata 如下：

Summary Data Set 1

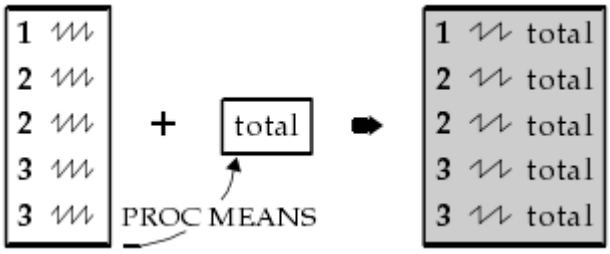
Obs	Exercise Type	_TYPE_	_FREQ_	Total
1	c-train	0	1	1190
2	running	0	2	6080
3	walking	0	3	5610

合并后的数据如下：

Sales Share by Type of Exercise 2

Exercise Type	Style	Sales	Total	Percent
c-train	Zip Sneak	1190	1190	100.000
running	Max Flight	1930	6080	31.743
	Zoom Airborne	4150	6080	68.257
walking	Zip Fit Leather	2250	5610	40.107
	Light Step	1130	5610	20.143
	Max Step Woven	2230	5610	39.750

6.7 合并 total 和原始数据



可以通过 means 过程创建一个包含总计（不是分组总计）的数据集。但不能直接与原始数据合并，因为没有匹配变量。幸运的是，可以用两个 set 语句：

```
DATA new-data-set;
    IF_N_=1 THEN SET summary-data-set;
    SET original-data-set;
```

原始数据包含了不止一个观测值，而 summary 数据集只有一个观测值。只在数据步的第一次迭代中，SAS 读取了 summary 数据集，之后为新数据的所有变量记住这个变量值。它的工作原理在于 SET 语句是自动记住的。往常之中，记住的变量会被下一个观测值改写，但这里变量只在第一次迭代的时候读取，并为所有观测值记住，这一技术适用于没有匹配变量的情况下，将一个单个观测值合并到多个观测值中。

例子 与上节同样的例子，现在想看每种类型的鞋子销量占总销量的百分比：

```
Max Flight       running 1930
Zip Fit Leather  walking 2250
Zoom Airborne    running 4150
Light Step       walking 1130
Max Step Woven   walking 2230
Zip Sneak        c-train 1190
```

代码为：

```
DATA shoes;
   INFILE 'c:\MyRawData\Shoesales.dat';
   INPUT Style $ 1-15 ExerciseType $ Sales;

* Output grand total of sales to a data set and print;
PROC MEANS NOPRINT DATA = shoes;
   VAR Sales;
   OUTPUT OUT = summarydata SUM(Sales) = GrandTotal;
PROC PRINT DATA = summarydata;
   TITLE 'Summary Data Set';

* Combine the grand total with the original data;
DATA shoesummary;
   IF _N_ = 1 THEN SET summarydata;
   SET shoes;
   Percent = Sales / GrandTotal * 100;
PROC PRINT DATA = shoesummary;
   VAR Style ExerciseType Sales GrandTotal Percent;
   TITLE 'Overall Sales Share';
RUN;
```

输出结果为：

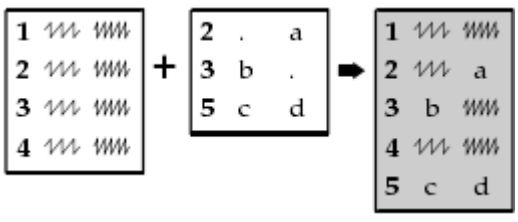
Summary Data Set 1

Obs	_TYPE_	_FREQ_	Grand Total
1	0	6	12880

Overall Sales Share 2

Obs	Style	Exercise Type	Sales	Grand Total	Percent
1	Max Flight	running	1930	12880	14.9845
2	Zip Fit Leather	walking	2250	12880	17.4689
3	Zoom Airborne	running	4150	12880	32.2205
4	Light Step	walking	1130	12880	8.7733
5	Max Step Woven	walking	2230	12880	17.3137
6	Zip Sneak	c-train	1190	12880	9.2391

6.8 用交易数据（transactions）更新主数据集（master）



update 语句用来更新大量新数据信息。

与 merge 语句一样，都是按照匹配变量来合并数据，不同点在于：

- 匹配变量的变量值有唯一性。
- 交易数据的缺失值不会改写主数据中存在的值。

基本形式为:

```
DATA master-data-set;
    UPDATE master-data-set transaction-data-set;
    BY variable-list;
```

只能指定两个数据集，一个是主数据集一个是交易数据，都需要按照匹配变量排序。且 BY 变量必须具有唯一性。

例子 一家医院有一份关于病人的主数据。变量依次为病人账户号码、名字、地址、出生日期、性别、保险代码、信息最后被更新的时间。

```
620135 Smith    234 Aspen St.      12-21-1975 m CBC 02-16-1998
645722 Miyamoto 65 3rd Ave.        04-03-1936 f MCR 05-30-1999
645739 Jensvold 505 Glendale Ave. 06-15-1960 f HLT 09-23-1993
874329 Kazoyan  76-C La Vista      .          . MCD 01-15-2003
```

当有新病人，或其他病人再进医院时，信息会被更新，比如，第一个病人的保险代码被更换了、最后一个病人的缺失数据被填补上、有新病人加入：

```
620135 .        .                  .          . HLT 06-15-2003
874329 .        .                  04-24-1954 m .   06-15-2003
235777 Harman   5656 Land Way      01-18-2000 f MCD 06-15-2003
```

下面的代码将这个数据放入一个名为 patientmaster 的永久数据集中，目录为 C 盘下的 MySASLib：

```
LIBNAME perm 'c:\MySASLib';
DATA perm.patientmaster;
   INFILE 'c:\MyRawData\Admit.dat';
   INPUT Account LastName $ 8-16 Address $ 17-34
      BirthDate MMDDYY10. Sex $ InsCode $ 48-50 @52 LastUpdate MMDDYY10.;
RUN;
```

下面的代码读取交易数据并排序，使用 updata 语句将交易数据更新到主数据中。

```
LIBNAME perm 'c:\MySASLib';
DATA transactions;
   INFILE 'c:\MyRawData\NewAdmit.dat';
   INPUT Account LastName $ 8-16 Address $ 17-34 BirthDate MMDDYY10.
      Sex $ InsCode $ 48-50 @52 LastUpdate MMDDYY10.;
PROC SORT DATA = transactions;
   BY Account;

* Update patient data with transactions;
DATA perm.patientmaster;
   UPDATE perm.patientmaster transactions;
   BY Account;
PROC PRINT DATA = perm.patientmaster;
   FORMAT BirthDate LastUpdate MMDDYY10.;
   TITLE 'Admissions Data';
RUN;
```

输出结果如下：

```
                          Admissions Data                          1

                                                          Ins
    Obs Account LastName Address            BirthDate Sex Code LastUpdate

     1  235777 Harman   5656 Land Way      01/18/2000  f  MCD  06/15/2003
     2  620135 Smith    234 Aspen St.      12/21/1975  m  HLT  06/15/2003
     3  645722 Miyamoto 65 3rd Ave.        04/03/1936  f  MCR  05/30/1999
     4  645739 Jensvold 505 Glendale Ave.  06/15/1960  f  HLT  09/23/1993
     5  874329 Kazoyan  76-C La Vista      04/24/1954  m  MCD  06/15/2003
```

6.9 使用 SAS 数据集选项

前面已经见过很多选项，SAS 语言主要有三种选项：系统选项、语句选项、数据集选项。系统选项有全局影响力，而数据集选项的影响力有限。

系统选项在 SAS 会话或工作期间都有效，包括 center 选项，它告诉 SAS，center 所有的输出。以及 LINESIZE=option，设置输出中每一行的最大长度。

语句选项出现在某个语句中，影响某一个数据步或者过程步。

数据集选项影响的只是 SAS 如何读取和写入一个单个的数据集，可以在数据步（DATA,SET,MERGE,or UPDATE 语句）和过程步（conjuction with a DATA=statement option）中使用。用法是，接在数据集名之后，用括号括起来。有些最常见的选项：

KEEP=variable-list	告诉 SAS 保留哪个变量
DROP=variable-list	告诉 SAS 丢弃哪个变量
RENAME=(oldvar=newvar)	重命名某个变量
FIRSTOBS=n	从观测值 n 开始读取变量
OBS=n	到观测值 n 停止读取
IN=new-var-name	

选择并重命名变量 下面是关于 KEEP=,DROP=，和 RENAME=的数据集选项的例子

```
DATA small;
   SET animals (KEEP = Cat Mouse Rabbit);

PROC PRINT DATA = animals (DROP = Cat Mouse Rabbit);

DATA animals (RENAME = (Cat = Feline Dog = Canine));
   SET animals;

PROC PRINT DATA = animals (RENAME =(Cat = Feline Dog = Canine));
```

KEEP=,DROP=，和 RENAME=的作用与 keep、drop、rename 很相似。区别在于，后者适用于数据步中的所有变量，而前者仅使用与语句前面的那个数据集。而且，后者仅可以在数据步中使用，而前者除了数据步和过程步，还可以在输入和输出数据集中使用。

用 observation number 选择观测值 可以使用 FIRSTOBS=和 obs=来选择读取哪些观测值

```
DATA animals;
   SET animals (FIRSTOBS = 101 OBS = 120);

PROC PRINT DATA = animals (FIRSTOBS = 101 OBS = 120);
```

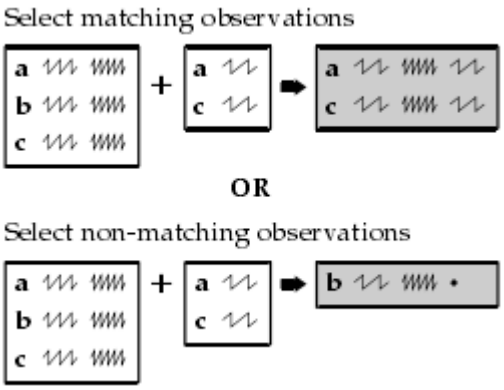
这也类似于同名的语句选项和同名的系统选项，语句选项只适用于 infile 语句，数据集选项是用于数据步和过程步中存在的数据集，而系统选项适用于所有的文件和数据集。如果同时使用同样的系统选项和数据集选项，那么后者将覆盖前者。

追踪观测值 这里提到的选项都是针对现有的变量，而 in=option 则自己创建一个新变量。这个新变量是临时的，并且有自己在选项中指定。下面的例子，SAS 创建了两个临时变量：InAnimals 和 InHabitat:

```
DATA animals;
   MERGE animals (IN = InAnimals) habitat (IN = InHabitat);
   BY Species;
```

该变量只存在于现在的过程步中。

6.10 用 in=option 追踪并选择观测值



如果合并了两个数据集，可以用 in=option 来追踪原始数据集对应新数据集中的哪个观测值。In=data 选项可以被用在数据步中的任何地方——SET,MERGE,或者 UPDATE——但大部分用在 merge 语句上，接在要追踪的数据集后面。

下面的数据步创建了一个 both 数据集，合并两个数据集，state 和 ounty。并用 in=Option 创建了两个变量 InState 和 InCounty：

```
DATA both;
   MERGE state (IN = InState) county (IN = InCounty);
   BY StateName;
```

创建的变量是临时的，只存在于数据步期间。SAS 给新变量赋值为 0 和 1。比如 county 数据集没有关于 Louisiana 的数据（Louisiana 只有 parishes，没有 counties），因此上例中，两个数据集都含有一个关于 Louisiana 的观测值，InState 变量下的这个观测值为 1，InCounty 的为 0。

这个被用在 IF 或 IF-THEN 语句中最多：

Subsetting IF: IF InState=1;
IF InCounty=0;
IF InState=1 AND InCounty=1;

IF-THEN: IF InCounty=1 THEN Origin=1;
IF InState=1 THEN State='Yes';

例子 一家运动品厂商有两份数据，一个包括所有客户的数据，一份包括了第三季度订单的数据。现在想要了解哪些客户在第三季度没有任何订单，即可以用 in=option 选项。客户数据包括客户编号、姓名、地址；订单数据包括客户编号、总价格：

Customer data

```
101 Murphy's Sports     115 Main St.
102 Sun N Ski           2106 Newberry Ave.
103 Sports Outfitters   19 Cary Way
104 Cramer & Johnson    4106 Arlington Blvd.
105 Sports Savers       2708 Broadway
```

Orders data

```
102 562.01
104 254.98
104 1642.00
101 3497.56
102 385.30
```

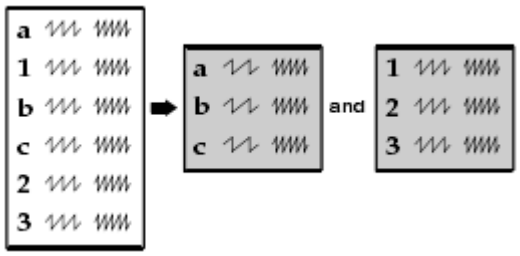
发现没有订单客户的代码如下，数据步中创建了新变量 recent，如果出现在客户数据中的观测值没有出现在 order 中，则 recent 赋为 0，否则赋为 1。

```
DATA customer;
   INFILE 'c:\MyRawData\Address.dat' TRUNCOVER;
   INPUT CustomerNumber Name $ 5-21 Address $ 23-42;
DATA orders;
   INFILE 'c:\MyRawData\OrdersQ3.dat';
   INPUT CustomerNumber Total;
PROC SORT DATA = orders;
   BY CustomerNumber;

* Combine the data sets using the IN= option;
DATA noorders;
   MERGE customer orders (IN = Recent);
   BY CustomerNumber;
   IF Recent = 0;
PROC PRINT DATA = noorders;
   TITLE 'Customers with No Orders in the Third Quarter';
RUN;
```

结果如下：

```
            Customers with No Orders in the Third Quarter            1

          Customer
    Obs    Number           Name              Address        Total

     1       103      Sports Outfitters    19 Cary Way          .
     2       105      Sports Savers        2708 Broadway        .
```

6.11 使用 output 语句写多维数据集



有时候想在一个数据步中创建多个数据，可以在 DATA 语句后面多接几个数据集名即可，如下语句告诉 SAS 创建三个数据集：LIONS、TIGERS、和 BEARS:

DATA lions tigers bears;

这样仅仅是创建了三个一样的数据集，如果想创建不同的，可以用 output 语句

每一个数据步的结尾都有一个暗含的 output 语句，它告诉 SAS 在处理下一个观测值之前，将当前的观测值写入输出数据集中。可以用自己的 output 语句来推翻这个暗含的 output 语句，基本形式为：

OUTPUT data-set-name;

如果遗漏了数据集名，则将被写入数据步中所有的数据集中去，output 可以单独使用，也可以使用在 IF-THEN 或 DO-loop 过程中：

IF family='Ursidae' THEN OUTPUT bears;

例子 有一份关于动物园给动物喂食的数据，变量为动物类型、生物学分类、居住区域、喂食是否在早上/下午/两者：

```
bears      Mammalia E2 both
elephants  Mammalia W3 am
flamingos  Aves     W1 pm
frogs      Amphibia S2 pm
kangaroos  Mammalia N4 am
lions      Mammalia W6 pm
snakes     Reptilia S1 pm
tigers     Mammalia W9 both
zebras     Mammalia W2 am
```

下面的代码创建了两个列表，一个是早上喂食，一个是下午喂食：

```
DATA morning afternoon;
   INFILE 'c:\MyRawData\Zoo.dat';
   INPUT Animal $ 1-9 Class $ 11-18 Enclosure $ FeedTime $;
   IF FeedTime = 'am' THEN OUTPUT morning;
      ELSE IF FeedTime = 'pm' THEN OUTPUT afternoon;
      ELSE IF FeedTime = 'both' THEN OUTPUT;
PROC PRINT DATA = morning;
   TITLE 'Animals with Morning Feedings';
PROC PRINT DATA = afternoon;
   TITLE 'Animals with Afternoon Feedings';
RUN;
```

日志：

```
NOTE: 9 records were read from the infile 'c:\MyRawData\Zoo.dat'.

NOTE: The data set WORK.MORNING has 5 observations and 4 variables.

NOTE: The data set WORK.AFTERNOON has 6 observations and 4 variables.
```

输出报告为：

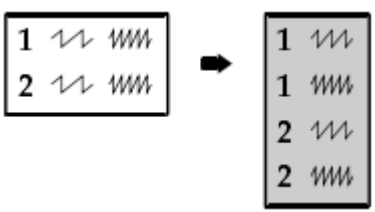
Animals with Morning Feedings 1

Obs	Animal	Class	Enclosure	Feed Time
1	bears	Mammalia	E2	both
2	elephants	Mammalia	W3	am
3	kangaroos	Mammalia	N4	am
4	tigers	Mammalia	W9	both
5	zebras	Mammalia	W2	am

Animals with Afternoon Feedings 2

Obs	Animal	Class	Enclosure	Feed Time
1	bears	Mammalia	E2	both
2	flamingos	Aves	W1	pm
3	frogs	Amphibia	S2	pm
4	lions	Mammalia	W6	pm
5	snakes	Reptilia	S1	pm
6	tigers	Mammalia	W9	both

6.12 使用 output 将一个观测值变成多个观测值



SAS 通常在数据步结尾将一个观测值写入数据中，但可以

写入多个观测值，在 DO loop 中或单独使用 output 语句。Output 语句控制何时将观测值写入 SAS 数据集中。如果数据集中没有 OUTPUT 语句，则暗含在结尾，放置了 output 之后，则结尾的就不在暗含存在。当 SAS 出现 OUTPUT 语句时，则写入一个观测值。

例子 下面的代码阐述如何在 DO LOOD 语句中使用 output 语句来产生一个数据集。

```
* Create data for variables x and y;
DATA generate;
   DO x = 1 TO 6;
      y = x ** 2;
      OUTPUT;
   END;
PROC PRINT DATA = generate;
   TITLE 'Generated Data';
RUN;
```

这个代码没有 INPUT 或 SET 语句，故整个数据步中只有一次迭代——但包括了 DO LOOP 中的六次循环。由于 OUTPUT 语句在 DO LOOP 循环中，因此每次循环都会创建一个观测值。如果没有 OUTPUT 语句，SAS 仅会写入一个观测值，因为结尾处暗含的 OUTPUT 语句：

```
                          Generated Data                          1

                          Obs    x     y

                           1     1     1
                           2     2     4
                           3     3     9
                           4     4    16
                           5     5    25
                           6     6    36
```

例子 有一份关于三个电影院的门票销售数据，记录了月份、电影院名称、门票销售额：

```
Jan Varsity 56723 Downtown 69831 Super-6 70025
Feb Varsity 62137 Downtown 43901 Super-6 81534
Mar Varsity 49982 Downtown 55783 Super-6 69800
```

现在需要将电影院名作为一个变量、销售额作为一个变量、月份重复三次。

下面的代码使用三次 input 语句读取同一个原始文件。第一个读取变量 month、location 和 tickets，并用@停留住数据行，接着用 OUTPUT 语句输出这个观测值。下一个 INPUT 读取这行后面的名、销售额，并再停留住行。接着读取，但释放行，进入下一个迭代。这个代码用 output 语句使每一行创建了三个观测值：

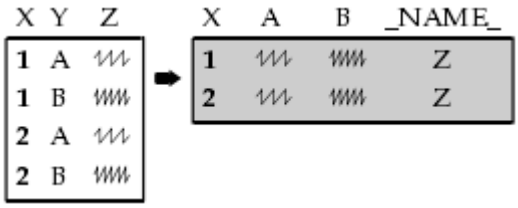
```
* Create three observations for each data line read
*   using three OUTPUT statements;
DATA theaters;
   INFILE 'c:\MyRawData\Movies.dat';
   INPUT Month $ Location $ Tickets @;
   OUTPUT;
   INPUT Location $ Tickets @;
   OUTPUT;
   INPUT Location $ Tickets;
   OUTPUT;
PROC PRINT DATA = theaters;
   TITLE 'Ticket Sales';
RUN;
```

结果为:

Ticket Sales 1

Obs	Month	Location	Tickets
1	Jan	Varsity	56723
2	Jan	Downtown	69831
3	Jan	Super-6	70025
4	Feb	Varsity	62137
5	Feb	Downtown	43901
6	Feb	Super-6	81534
7	Mar	Varsity	49982
8	Mar	Downtown	55783
9	Mar	Super-6	69800

6.13 使用 proc transpose 将观测值转变为变量



transpose 过程可以转置数据集，将观测值转变为变量或将变量转变为观测值。不部分情况下，将观测值转变为变量，可以使用下面代码：

```
PROC TRANSPOSE DATA=old-data-set OUT=new-data-set;
BY variable-list;
ID variable;
VAR variable-list;
```

BY 语句

ID 语句 ID 语句命名变量，这些变量值将变成新的变量名，ID 变量在一个数据集中只能发生一次，如果有 BY 语句，那么在一个 by-group 中，变量值必须是唯一的。如果 ID 变量是数值型变量，新的变量名必须有一个下划线作为前缀（_1 or_2,for example）。如果不适用 ID 语句，新变量将命名为 COL1，COL2 等。

VAR 语句 VAR 语句命名变量，这个变量的变量值是要转置的。

例子 有一份关于儿童棒球联盟选手的数据，包括队名、选手编号、数据类型(salary or batting average)、以及 entry：

```
Garlics 10 salary 43000
Peaches  8 salary 38000
Garlics 21 salary 51000
Peaches 10 salary 47500
Garlics 10 batavg .281
Peaches  8 batavg .252
Garlics 21 batavg .265
Peaches 10 batavg .301
```

现在想考察平均打击数与 salary 之间的关系，首先要将 salary 和平均打击数变量变量。下面的代码读取数据、按照队伍、选手排序数据，再转置数据：

```
DATA baseball;
   INFILE 'c:\MyRawData\Transpos.dat';
   INPUT Team $ Player Type $ Entry;
PROC SORT DATA = baseball;
   BY Team Player;
PROC PRINT DATA = baseball;
   TITLE 'Baseball Data After Sorting and Before Transposing';

* Transpose data so salary and batavg are variables;
PROC TRANSPOSE DATA = baseball OUT = flipped;
   BY Team Player;
   ID Type;
   VAR Entry;
PROC PRINT DATA = flipped;
   TITLE 'Baseball Data After Transposing';
RUN;
```

在 proc transpose 这步中，BY 变量是 team 和 player。ID 变量是 type，它的值 salary 和 batavg 将是新变量名，将要转置的变量 entry 在 VAR 语句中指定。注意原来是变量名的 entry，现在作为_name_变量下面的变量值。结果为：

Baseball Data After Sorting and Before Transposing 1

Obs	Team	Player	Type	Entry
1	Garlics	10	salary	43000.00
2	Garlics	10	batavg	0.28
3	Garlics	21	salary	51000.00
4	Garlics	21	batavg	0.27
5	Peaches	8	salary	38000.00
6	Peaches	8	batavg	0.25
7	Peaches	10	salary	47500.00
8	Peaches	10	batavg	0.30

Baseball Data After Transposing 2

Obs	Team	Player	_NAME_	salary	batavg
1	Garlics	10	Entry	43000	0.281
2	Garlics	21	Entry	51000	0.265
3	Peaches	8	Entry	38000	0.252
4	Peaches	10	Entry	47500	0.301

6.14 使用 SAS 自动变量

SAS 有一些自动变量，这些变量看不到，是临时并不会被储存。但在数据步中，可以任意使用。

_N_和_ERROR_ _N_记录了 SAS 在数据步中循环的次数，它不一定等于循环次数。因为诸如 IF 语句就可以使迭代次数与观测数不一致。如果一个观测值的数据出现错误，_ERROR_会被赋值为 1，否则赋值为 0。错误数据包括无效数据（数值型格式变量却赋为字符串值），转换错误（0 作为除数），函数中不合法的自变量（log（0））。

FIRST.variable 和 LAST.variable 当使用 BY 语句时，这两个变量才有效。SAS 处理一个观测值时，如果某个变量的新变量值是第一次出现，first.variable 被赋值为 1，其他观测值中被赋为 0。LAST.variable 是同理的某变量的变量值是最后一次出现时，赋为 1，其他赋为 0。

例子 有一份不知道绕着镇中心走路比赛的数据，变量为 entry number、age group 和 finishing time。注意每行不止一个观测值：

```
54 youth  35.5 21 adult  21.6  6 adult  25.8 13 senior 29.0
38 senior 40.3 19 youth  39.6  3 adult  19.0 25 youth  47.3
11 adult  21.9  8 senior 54.3 41 adult  43.0 32 youth  38.6
```

第一件要做的事情是为完成情况创建一个新变量。下面代码读取数据，按照 finishing time 排序，另一个数据步创建新变量 place，并赋给它当前的_N_值，print 过程产生 finishers 列表：

```
DATA walkers;
   INFILE 'c:\MyRawData\Walk.dat';
   INPUT Entry AgeGroup $ Time @@;
PROC SORT DATA = walkers;
   BY Time;
* Create a new variable, Place;
DATA ordered;
   SET walkers;
   Place = _N_;
PROC PRINT DATA = ordered;
  TITLE 'Results of Walk';

PROC SORT DATA = ordered;
   BY AgeGroup Time;
* Keep the first observation in each age group;
DATA winners;
   SET ordered;
   BY AgeGroup;
   IF FIRST.AgeGroup = 1;
PROC PRINT DATA = winners;
   TITLE 'Winners in Each Age Group';
RUN;
```

第二段代码给出了每个年龄组的第一名：BY 语句中自动产生了 first.variable，后面的 IF 语句保留了每个年龄组的第一个观测值，由于数据是按照年龄组 agegroup 和 time 排序的，因此第一个观测值就是第一名。结果为：

Results of Walk 1

Obs	Entry	Age Group	Time	Place
1	3	adult	19.0	1
2	21	adult	21.6	2
3	11	adult	21.9	3
4	6	adult	25.8	4
5	13	senior	29.0	5
6	54	youth	35.5	6
7	32	youth	38.6	7
8	19	youth	39.6	8
9	38	senior	40.3	9
10	41	adult	43.0	10
11	25	youth	47.3	11
12	8	senior	54.3	12

Winners in Each Age Group 2

Obs	Entry	Age Group	Time	Place
1	3	adult	19.0	1
2	13	senior	29.0	5
3	54	youth	35.5	6

第七章 使用 SAS 宏功能灵活写代码

7.1 宏概述

宏以前被认为是经验老道的 SAS 程序员使用的高级技术。但现在新手也能够了解一些。幸运的是，宏的基本功能不是那么难理解。

本章介绍的最普遍使用的 SAS 宏语言的特征。

宏处理器 标准 SAS 程序中，提交程序后，SAS 就编译并立即执行。但宏语句多了一步，在提交之后，SAS 会将宏语句传送到宏处理器上，将其转变为 SAS 标准代码，通常叫做“变换代码”（meta-programming.）

宏和宏变量 SAS 宏代码包括两个基本部分：宏命令和宏变量。宏变量通常加一个“&”作为前缀，而宏命令通常加一个“%”作为前缀。

局部 VS 全局 宏变量有局部宏变量和全局宏变量。如果在宏的内部定义则为局部宏变量，只能在内部使用。如果在开放代码中定义则为全局宏变量。避免两种错误：在宏之外（开放代码）使用局部变量；创建同名的局部变量和全局变量。

启动宏处理器 使用宏指令之前必须将宏系统选项打开，尽管有时默认是打开的。可以用下面代码查看是否打开： PROC OPTIONS OPTION=MACRO; RUN;

查看日志，如果看到 MACRO，则打开了；如果看到 NOMACRO，则没有打开。

避免宏错误 宏会让人很头痛，可以通过分段形式避免。首先，用 SAS 标准语言写下程序；接着，将其转变为宏代码。

7.2 用宏变量提交文本

用%let 创建一个宏变量 最简单给宏变量分配一个值的方法是%let，基本形式为：

%LET macro-variable-name=value;

宏变量名必须符合 SAS 命名法则，（少于等于 32 字节、以字母或下划线开头、只能包括字母、数字和下划线），下面创建了宏变量：

```
%LET iterations=10;
%LET country=New Zealand;
```

当赋值字符串时，不需要加引号。除非开头和结尾的空格，否则从等号到分号的全部内容都是变量值。

使用宏变量 宏变量前面要加前缀&，注意宏处理器找不到单引号内的宏变量，只能用双引号。如下的例子：

```
DO    i=1 to &iterations;
TITLE ”Addresses in &country”;
```

经宏处理器编译后，语句变成：

```
DO    i=1 to 10;
TITLE ”Addresses in New Zealand”;
```

例子 一份关于花店销售的数据，变量为顾客 ID，销售日期，花的种类，数量：

```
240W 02-07-2003 Ginger    120
240W 02-07-2003 Protea    180
356W 02-08-2003 Heliconia  60
356W 02-08-2003 Anthurium 300
188R 02-11-2003 Ginger     24
188R 02-11-2003 Anthurium  24
240W 02-12-2003 Heliconia  48
240W 02-12-2003 Protea     48
356W 02-12-2003 Ginger    240
```

想要一份反映某一种类型花的销售情况数据，宏变量就可以不用编辑数据步和过程步来选择一种花种：

```
%LET flowertype = Ginger;

* Read the data and subset with a macro variable;
DATA flowersales;
   INFILE 'c:\MyRawData\TropicalSales.dat';
   INPUT CustomerID $ @6 SaleDate MMDDYY10.
      @17 Variety $9. Quantity;
   IF Variety = "&flowertype";

* Print the report using a macro variable;
PROC PRINT DATA = flowersales;
   FORMAT SaleDate WORDDATE18.;
   TITLE "Sales of &flowertype";
RUN;
```

结果如下：

```
                          Sales of Ginger                        1

          Customer
 Obs        ID                   SaleDate     Variety    Quantity

  1        240W         February 7, 2003      Ginger         120
  2        188R        February 11, 2003      Ginger          24
  3        356W        February 12, 2003      Ginger         240
```

7.3 用宏创建一个模块化的代码

宏可以使一段代码在一个或多个程序中被反复使用，而不需要重复的去编写相同或相似的代码。基本形式为：

```
%MACRO macro-name;
    macro-text
%MEND macro-name;
```

%MACRO 语句告诉 SAS 这是宏开始，而%MEND 则意味着结束。macro-name 是自己命名的，但 mend 后面的 macro-name 则是可选的，但加入会使得句子好很多（easier to debug and maintain）。

启动宏 定义了宏之后，可以通过在宏名称前面增加%来启动宏：%macro-name，注意这里可以不用分号。

例子 使用前面花店销售的数据：

```
240W 02-07-2003 Ginger    120
240W 02-07-2003 Protea    180
356W 02-08-2003 Heliconia  60
356W 02-08-2003 Anthurium 300
188R 02-11-2003 Ginger     24
188R 02-11-2003 Anthurium  24
240W 02-12-2003 Heliconia  48
240W 02-12-2003 Protea     48
356W 02-12-2003 Ginger    240
```

下面的代码创建了一个名为%SAMPLE 的宏，来将数据按照 Quantity 排序，打印出最大销售额的五个观测值。然后用标准数据步读取数据，并启动宏：

```
* Macro to print 5 largest sales;
%MACRO sample;
   PROC SORT DATA = flowersales;
      BY DESCENDING Quantity;
   PROC PRINT DATA = flowersales (OBS = 5);
      FORMAT SaleDate WORDDATE18.;
      TITLE 'Five Largest Sales';
%MEND sample;

* Read the flower sales data;
DATA flowersales;
   INFILE 'c:\MyRawData\TropicalSales.dat';
   INPUT CustomerID $ @6 SaleDate MMDDYY10. @17
      Variety $9. Quantity;
RUN;

* Invoke the macro;
%sample
RUN;
```

结果为：

Five Largest Sales 1

Obs	Customer ID	SaleDate	Variety	Quantity
1	356W	February 8, 2003	Anthurium	300
2	356W	February 12, 2003	Ginger	240
3	240W	February 7, 2003	Protea	180
4	240W	February 7, 2003	Ginger	120
5	356W	February 8, 2003	Heliconia	60

这样的宏有些限制，因为它只能做一件事。现在结合%let 语句，并增加参数使得其更加灵活

宏自动呼叫库 本书中的宏仅在一个程序内部定义和启动。也可以将宏储存在一个中心位置，叫做自动呼叫库，被各个程序分享。具体来说，将宏作为文件储存在某路径中，或作为分区数据集中的一员。使用 MAUTOSOURCE 和 SASAUTOS=系统选项告诉 SAS 在哪里查找宏。之后，即使宏没有出现在程序中，也可以启动它了

7.4 给宏增加参数

参数就是宏的变量，给宏增加参数，在%MACRO 语句中的括号内列出宏变量的名字。基本形式为：

```
%MACRO macro-name(parameter-1=,parameter-2=,...parameter-n=);
      macro-text
```

%MEND macro-name;

比如，一个叫做%QUARTERLYREPORT 的宏可能这样开始：

%MACRO quarterlyreport(quarter=,salesrep=);

这个宏有两个参数&QUARTER 和&SALESREP。启用这个宏可以用这样的语句：

%quarterlyreport(quarter=3,salesrep=Smith)

例子 仍然是花店销售例子：

```
240W 02-07-2003 Ginger    120
240W 02-07-2003 Protea    180
356W 02-08-2003 Heliconia  60
356W 02-08-2003 Anthurium 300
188R 02-11-2003 Ginger     24
188R 02-11-2003 Anthurium  24
240W 02-12-2003 Heliconia  48
240W 02-12-2003 Protea     48
356W 02-12-2003 Ginger    240
```

现在需要一份报告，反映每位对每位顾客的销售。下面的代码定义了宏%SELECT，并启动两次。这个宏排序并打印数据 FlowerSales，使用参数创造了两位命为&CUSTOMER 和&SORTVAR 的宏变量：

```
* Macro with parameters;
%MACRO select(customer=,sortvar=);
   PROC SORT DATA = flowersales OUT = salesout;
      BY &sortvar;
      WHERE CustomerID = "&customer";
   PROC PRINT DATA = salesout;
      FORMAT SaleDate WORDDATE18.;
      TITLE1 "Orders for Customer Number &customer";
      TITLE2 "Sorted by &sortvar";
%MEND select;

 * Read all the flower sales data;
 DATA flowersales;
    INFILE 'c:\MyRawData\TropicalSales.dat';
    INPUT CustomerID $ @6 SaleDate MMDDYY10. @17
       Variety $9. Quantity;
 RUN;

*Invoke the macro;
%select(customer = 356W, sortvar = Quantity)
%select(customer = 240W, sortvar = Variety)
RUN;
```

结果为：

Orders for Customer Number 356W 1
Sorted by Quantity

Obs	Customer ID	SaleDate	Variety	Quantity
1	356W	February 8, 2003	Heliconia	60
2	356W	February 12, 2003	Ginger	240
3	356W	February 8, 2003	Anthurium	300

Orders for Customer Number 240W 2
Sorted by Variety

Obs	Customer ID	SaleDate	Variety	Quantity
1	240W	February 7, 2003	Ginger	120
2	240W	February 12, 2003	Heliconia	48
3	240W	February 7, 2003	Protea	180
4	240W	February 12, 2003	Protea	48

7.5 使用条件逻辑写宏代码

在宏中使用条件逻辑的基本形式为：

```
%IF condition%THEN action;
    %ELSE%IF condition%THEN action;
    %ELSE action;
%IF condition%THEN%DO;
    SAS statements
%END;
```

自动宏变量 每一次启动 SAS，宏处理器自动创建一些宏变量，可以使用在程序中。最常见的有：

Variable name	Example	Description
&SYSDATE	29MAY02	the character value of the date that job or session began
&SYSDAY	Wednesday	the day of the week that job or session began

比如，可以像这样结合自动宏变量和条件逻辑语句：

```
%IF  &SYSDAY=Tuesday  %THEN  %LET country=Belgium;
%ELSE  %LET country=France;
```

例子 仍然是花店销售数据：

```
240W 02-07-2003 Ginger    120
240W 02-07-2003 Protea    180
356W 02-08-2003 Heliconia  60
356W 02-08-2003 Anthurium 300
188R 02-11-2003 Ginger     24
188R 02-11-2003 Anthurium  24
240W 02-12-2003 Heliconia  48
240W 02-12-2003 Protea     48
356W 02-12-2003 Ginger    240
```

现在想在周一和周四的时候打印报告，代码如下：

```
%MACRO dailyreports;
   %IF &SYSDAY = Monday %THEN %DO;
      PROC PRINT DATA = flowersales;
         FORMAT SaleDate WORDDATE18.;
         TITLE 'Monday Report: Current Flower Sales';
   %END;
   %ELSE %IF &SYSDAY = Tuesday %THEN %DO;
      PROC MEANS DATA = flowersales MEAN MIN MAX;
         CLASS Variety;
         VAR Quantity;
         TITLE 'Tuesday Report: Summary of Flower Sales';
   %END;
%MEND dailyreports;

DATA flowersales;
   INFILE 'c:\MyRawData\TropicalSales.dat';
   INPUT CustomerID $ @6 SaleDate MMDDYY10. @17
      Variety $9. Quantity;
RUN;

%dailyreports
RUN;
```

当程序提交之后，宏处理器将会写下如下代码：

```
DATA flowersales;
   INFILE 'c:\MyRawData\TropicalSales.dat';
   INPUT CustomerID $ @6 SaleDate MMDDYY10. @17
      Variety $9. Quantity;
RUN;

PROC MEANS DATA = flowersales MEAN MIN MAX;
   CLASS Variety;
   VAR Quantity;
   TITLE 'Tuesday Report: Summary of Flower Sales';
RUN;
```

如果这段代码是周四写的，那么结果如下：

Tuesday Report: Summary of Flower Sales 1

The MEANS Procedure
Analysis Variable : Quantity

Variety	N Obs	Mean	Minimum	Maximum
Anthurium	2	162.0000000	24.0000000	300.0000000
Ginger	3	128.0000000	24.0000000	240.0000000
Heliconia	2	54.0000000	48.0000000	60.0000000
Protea	2	118.0000000	48.0000000	180.0000000

7.6 用 call symput 编写数据驱动的程序

分配一个值给宏变量，可以使用 call symput：

CALL SYMPUT(”macro-variable-name”,value);

macro-variable-name 是变量名，value 可以是一个变量名，该变量的值将分配给 macro-variable-name，也可以是一个用引号括起来的常量。

call symput 通常在 IF-THEN 语句中使用：

```
IF Age>=18 THEN CALL SYMPUT(”status”,”Adult”);
  ELSE CALL SYMPUT(”status”,”Minor”);
```

这个语句创建了宏变量&STATUS，并依据年龄情况分配给值 adult 或 minor。下面的代码使用变量来赋值：

```
IF TotalSales>1000000 THEN CALL SYMPUT(”bestseller”,BookTitle);
```

注意 用 call symput 创建的宏变量与赋值变量不能够用在同一个数据步中。因为直到数据步执行之后，SAS 才会将一个值赋给宏变量。

例子 仍然是花店销售的数据：

```
240W 02-07-2003 Ginger    120
240W 02-07-2003 Protea    180
356W 02-08-2003 Heliconia  60
356W 02-08-2003 Anthurium 300
188R 02-11-2003 Ginger     24
188R 02-11-2003 Anthurium  24
240W 02-12-2003 Heliconia  48
240W 02-12-2003 Protea     48
356W 02-12-2003 Ginger    240
```

现在想找到单个订单最大的客户，并打印出这个客户的所有订单。

```
* Read the raw data;
DATA flowersales;
   INFILE 'c:\MySASLib\TropicalSales.dat';
   INPUT CustomerID $4. @6 SaleDate MMDDYY10. @17
      Variety $9. Quantity;
PROC SORT DATA = flowersales;
   BY DESCENDING Quantity;

* Find biggest order and pass the customer id to a macro variable;
DATA _NULL_;
   SET flowersales;
   IF _N_ = 1 THEN CALL SYMPUT("selectedcustomer",CustomerID);
   ELSE STOP;

PROC PRINT DATA = flowersales;
   WHERE CustomerID = "&selectedcustomer";
   FORMAT SaleDate WORDDATE18.;
   TITLE "Customer &selectedcustomer Had the Single Largest Order";
RUN;
```

第一段代码读取原始数据，proc sort 按照 quantity 降序排序，最大单个订单将会出现在第一个观测值上。

第二段代码使用 call symput，当_N_为 1 的时候，分配变量 CustomerID 的值给宏变量&SELECTEDCUSTOMER，在数据步中，我们所需要的就是这么多，因此使用 stop 语句告诉 SAS 停止数据步。Stop 语句也可以不要，但为了提高效率，它可以告诉 SAS 不要再读取下面的观测值了。

第三段代码，此时 SAS 直到数据步已经结束，因此执行数据步。宏变量&SELECTEDCUSTOMER 的值为 356W，结果如下：

Customer 356W Had the Single Largest Order 1

Obs	Customer ID	SaleDate	Variety	Quantity
1	356W	February 8, 2003	Anthurium	300
2	356W	February 12, 2003	Ginger	240
5	356W	February 8, 2003	Heliconia	60

7.7 排除宏错误的 bug

避免宏错误 尽可能先用标准 SAS 代码写你的程序，当没有错误了，再转成宏代码，先一次增加一个宏逻辑特征。再增加%macro 和%mend。再增加宏变量。

引用问题 宏处理器不能解决但引号内的宏。所以要使用双引号。比如下面的例子，单引号不能读取宏变量的值：

Original statement	Statement after resolution
TITLE 'Report for &month';	TITLE 'Report for &month';
TITLE "Report for &month";	TITLE "Report for January";

排除宏错误 bug 的系统选项 这五个系统选项会影响 SAS 写入日志的信息。粗体为默认的设置。

MERROR|NOMERROR 如果你调用了一个 SAS 不能找到的宏，则会报警。

SERROR|NOSERROR 如果你使用了一个 SAS 不能找到的宏，则会报警。

MLOGIC|**NOMLOGIC** SAS 会在日志里打印关于执行宏的详细信息。

MPRINT|**NOMPRINT** SAS 在日志里打印由宏产生的标准 SAS 代码。

SYMBOLGEN|**NOSYMBOLGEN** SAS 在日志里打印宏变量的值。

最好只在排除 bug 的时候才将 MLOGIC,MPRINT 和 SYMBOLGEN 打开，否则它会让你的日志阅读起来很困难。想要关闭，则用系统语句：

```
OPTIONS MPRINT NOSYMBOLGEN NOMLOGIC;
```

Merror 信息 如果 SAS 不能找到一个宏，并且 Merror 选项也是开着的，那么 SAS 会打印这样的信息：

```
WARNING:Apparent invocation of macro SAMPL not resolved.
```

确认宏名字的拼写是否正确。

SERROR 信息 如果 SAS 不能在开放代码中处理一个宏变量，并且 serror 选项是开着的，SAS 会打印这样的信息：

```
WARNING:Apparent symbolic reference FLOWER not resolved.
```

首先确认是否拼写错误，再次查看视角，即是否在外部使用了一个局部变量。

MLOGIC 信息 如果这个选项开启，SAS 会在日志中打印由宏产生的 SAS 语句。如果在 MPRINT 选项中运行了%SAMPLE，日志会如下所示;

```
36    OPTIONS MPRINT;
37    %sample(flowertype=Anthurium)
MPRINT(SAMPLE):    PROC PRINT DATA = flowersales;
MPRINT(SAMPLE):    WHERE Variety = "Anthurium";
MPRINT(SAMPLE):    RUN;
```

SYMBOLGEN 信息 如果这个选项开启，SAS 会在日志窗口中打印每个宏变量的值。如果在 SYMBOLGEN 选项中运行%SAMPLE，日志会如下所示：

```
30    OPTIONS SYMBOLGEN;
31    %sample(flowertype=Anthurium)
SYMBOLGEN:  Macro variable FLOWERTYPE resolves to Anthurium
```

第八章 使用基本统计过程

8.1 用 PROC UNIVARIATE 检验数据分布

PROC UNIVARIATE 是 Base SAS software 的一部分，产生统计量以描述单个变量的分布。这些统计量包括均值、中位数、mode、标准差、偏度和峰度。

Proc UNIVARIATE 的使用很简单，在 proc 语句之后，用 var 语句指定一个或多个变量：

```
PROC UNIVARIATE;
    VAR variable-list;
```

没有 var 语句，SAS 会计算所有数值变量的统计量。Proc 语句中也可以指定其他选项，比如 plot 或 normal：

```
PROC UNIVARIATE PLOT NORMAL;
```

Normal 选项进行正态测试，PLOT 画出数据的三个图(stem-and-leaf plot,box plot,and normal probability plot)，可以使用 BY 语句来对单个组进行分析。（前提是要进行 sort 排序）

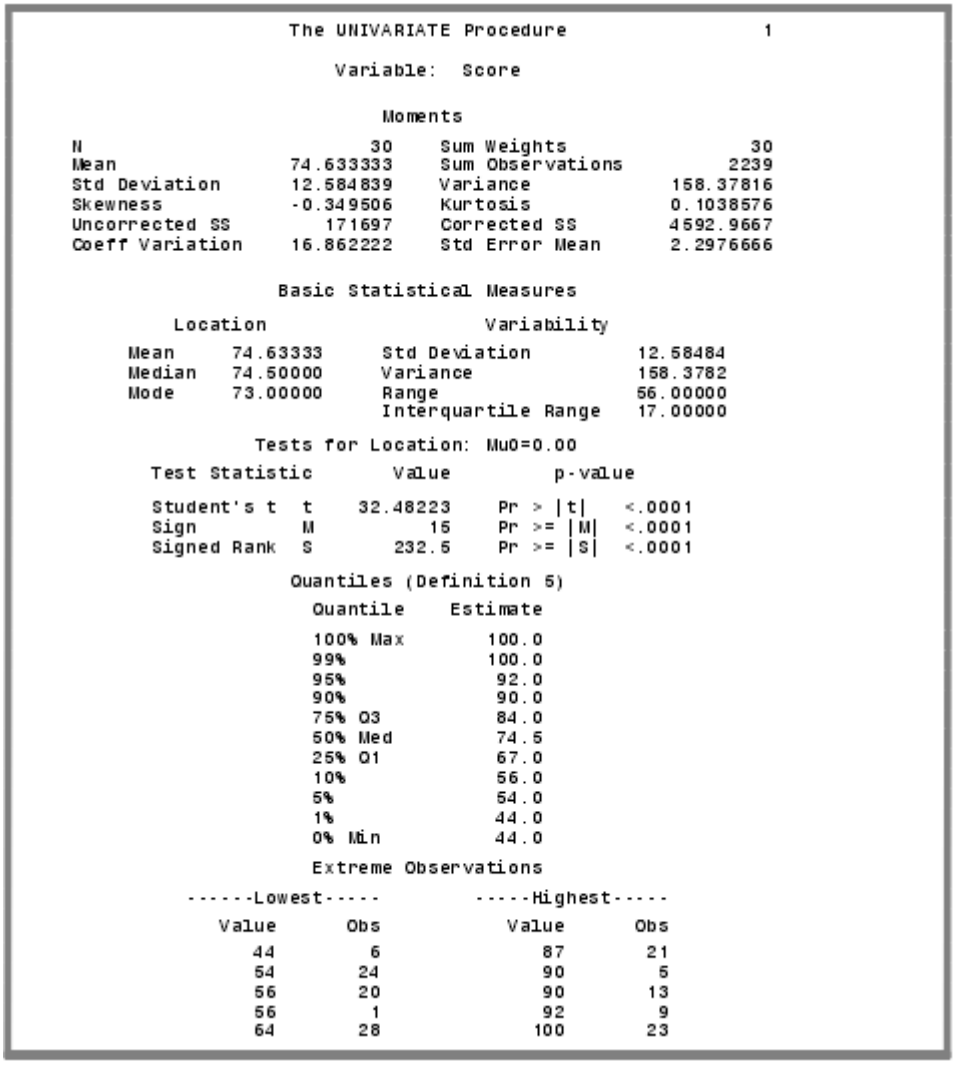
例子 下面的数据是一个班级的学生分数,每一行是 10 位同学的分数:

```
56 78 84 73 90 44 76 87 92 75
85 67 90 84 74 64 73 78 69 56
87 73 100 54 81 78 69 64 73 65
```

下面代码读取数据并运行 PROC UNIVARIATE:

```
DATA class;
   INFILE 'c:\MyRawData\Scores.dat';
   INPUT Score @@;
PROC UNIVARIATE DATA = class;
   VAR Score;
   TITLE;
RUN;
```

结果为:



The UNIVARIATE Procedure 1

Variable: Score

Moments

N	30	Sum Weights	30
Mean	74.633333	Sum Observations	2239
Std Deviation	12.584839	Variance	158.37816
Skewness	-0.349506	Kurtosis	0.1038576
Uncorrected SS	171697	Corrected SS	4592.9667
Coeff Variation	16.862222	Std Error Mean	2.2976666

Basic Statistical Measures

Location		Variability	
Mean	74.63333	Std Deviation	12.58484
Median	74.50000	Variance	158.3782
Mode	73.00000	Range	56.00000
		Interquartile Range	17.00000

Tests for Location: Mu0=0.00

Test	Statistic	Value	p-value
Student's t	t	32.48223	Pr > |t| <.0001
Sign	M	15	Pr >= |M| <.0001
Signed Rank	S	232.5	Pr >= |S| <.0001

Quantiles (Definition 5)

Quantile	Estimate
100% Max	100.0
99%	100.0
95%	92.0
90%	90.0
75% Q3	84.0
50% Med	74.5
25% Q1	67.0
10%	56.0
5%	54.0
1%	44.0
0% Min	44.0

Extreme Observations

------Lowest-----		-----Highest-----	
Value	Obs	Value	Obs
44	6	87	21
54	24	90	5
56	20	90	13
56	1	92	9
64	28	100	23

8.2 用 proc means 产生统计量

由 Proc univariate 产生的统计量，大部分都可以由 proc means 产生，前提是你要要求它产生。Proc univariate 会默认打印所有的统计量：mean,variance,skewness,quantiles,extremes,t tests,standard error。而用 proc means 你可以要求打印你需要的统计量。

Means 过程只需要一个语句：

PROC MEANS statistic-keywords;

默认 means 会产生均值、缺失值数、标准差、每一个数值变量的最小最大值，下面的 list 列出可以需要的统计量，可以将它们加入 proc mean 语句中：

CLM	双侧置信区间	RANGE	范围
CSS	调整的平方和	SKEWNESS	偏度
CV	变异系数	STDDEV	标准差
KURTOSIS	峰度	STDERR	平均标准差
LCLM	单侧（低侧）置信区间	SUM	总和
MAX	最大值	SUMWGT	权数变量之和
MEAN	平均数	UCLM	单侧（高侧）置信区间
MIN	最小值	USS	未调整的平方和

N 非缺失变量值的个数　　VAR 方差?

NMISS 缺失值变量个数　　PROBT probability for Student's t

MEDIAN 中位数　　T Student's t

Q1(P25)25%quantile　　Q3(P75)75%quantile

P1 1%quantile　　P5 5%quantile

P10 10%quantile　　P90 90%quantile

P95 95%quantile　　P99 99%quantile

置信区间 默认置信区间的置信水平为 0.05 或 95%，用在 means 语句中使用 ALPHA=option 可以得到不同的置信度。比如现在想要 90%的置信区间，就要指定 ALPHA=0.10，并要有 clm 选项，语句为：

```
PROC MEANS ALPHA=.10 CLM;
```

VAR 语句 means 会默认为所有的数值型变量产生统计量，如果不需要，那么用 var 语句中指定你需要的变量，基本形式：

```
PROC MEANS options;
  VAR variable-list;
```

例子 如下是书店中关于儿童读物的书本页数：

```
34 30 29 32 52 25 24 27 31 29
24 26 30 30 30 29 21 30 25 28
28 28 29 38 28 29 24 24 29 31
30 27 45 30 22 16 29 14 16 29
32 20 20 15 28 28 29 31 29 36
```

Means 可以产生平均页数及 90%的置信区间：

```
DATA booklengths;
   INFILE 'c:\MyRawData\Picbooks.dat';
   INPUT NumberOfPages @@;
*Produce summary statistics;
PROC MEANS DATA=booklengths N MEAN MEDIAN CLM ALPHA=.10;
   TITLE 'Summary of Picture Book Lengths';
RUN;
```

结果为：

```
                    Summary of Picture Book Lengths                1

                          The MEANS Procedure

                   Analysis Variable : NumberOfPages

                                              Lower 90.0%     Upper 90.0%
 N            Mean          Median           CL for Mean     CL for Mean
-------------------------------------------------------------------------
50      28.0000000      29.0000000       26.4419136      29.5580864
-------------------------------------------------------------------------
```

8.3 用 proc freq 检验分类数据

PROC FREQ,是 base SAS 的一部分，可以产生很多统计量来检验分类数据的相关性。基本形式为：

```
PROC FREQ;
 TABLES variable-combinations/options;
```

选项 这里有一些统计选项：

AGREE 检测分类性，包括 McNemar’s test,Bowker’s test,Cochran’s Q test,and kappa statistics

CHISQ 用卡方统计量检测一致性和同类性。

CL 一致性检测的置信区间

CMH Cochran-Mantel-Haenszel statistics

EXACT Fisher’s exact test for tables larger than 2X2

MEASURES 一致性测量，包括 Pearson and Spearman correlation coefficients,gamma,Kendall’s tau-b,Stuart’s tau-c,Somer’s D。

PLCORR polychoric correlation coefficient

RELRISK relative risk measures for 2X2 tables

TREND the Cochran-Armitage test for trend

例子 有人抱怨公交班车总是到站太慢，而快速巴士却可以准时到达。现在想弄明白车的种类与是否准时之间的关系。现在有一组数据，包括两个变量：车类型（E for express or R for regular），是否准时（L for late or O for on time），每一行包含 10 个观测值：

```
E O E L E L R O E O E O E O R L R O R L
R O E O R L E O R L R O E O E O R L E L
E O R L E O R L E O R L E O R O E L E O
E O E O E O E L E O E O R L R L R O R L
E L E O R L R O E O E O E O E L R O R L
```

下面的代码读取数据，用 chisq 选项运行 proc freq，

```
DATA bus;
   INFILE 'c:\MyRawData\Bus.dat';
   INPUT BusType $  OnTimeOrLate $ @@;
PROC FREQ DATA = bus;
   TABLES BusType * OnTimeOrLate / CHISQ;
   TITLE;
RUN;
```

结果为：

The FREQ Procedure

Table of BusType by OnTimeOrLate

BusType	OnTimeOrLate		
Frequency Percent Row Pct Col Pct	L	O	Total
E	7 14.00 24.14 35.00	22 44.00 75.86 73.33	29 58.00
R	13 26.00 61.90 65.00	8 16.00 38.10 26.67	21 42.00
Total	20 40.00	30 60.00	50 100.00

Statistics for Table of BusType by OnTimeOrLate

Statistic	DF	Value	Prob
Chi-Square	1	7.2386	0.0071
Likelihood Ratio Chi-Square	1	7.3364	0.0068
Continuity Adj. Chi-Square	1	5.7505	0.0165
Mantel-Haenszel Chi-Square	1	7.0939	0.0077
Phi Coefficient		-0.3805	
Contingency Coefficient		0.3556	
Cramer's V		-0.3805	

Fisher's Exact Test

Cell (1,1) Frequency (F)	7
Left-sided Pr <= F	0.0081
Right-sided Pr >= F	0.9987
Table Probability (P)	0.0067
Two-sided Pr <= P	0.0097

Sample Size = 50

8.4 用 proc corr 检测相关性

基本形式为：PROC CORR；

它告诉 SAS 计算最近创建的数据集中的所有数值变量两两相关系数。可以和 VAR 和 with 来指定变量：

VAR variable-list;

WITH variable-list;

VAR 语句中的变量出现在交叉表顶部，而 with 的变量出现在左侧。

默认情况下，proc corr 计算 Pearson 积差相关系数。可以增加选项要求非参数的相关系数。下面的 SPEARMAN 选项告诉 SAS 计算 Spearman’s rank correlations，而不是 Pearson’s

correlations：

PROC CORR SPEARMAN;

其他还有如 HOEFFDING(for Hoeffding's D statistic) 何 KENDALL(for Kendall'stau-b coefficients)

例子 有一份关于学生学习的数据，变量为考试分数，一周花在电视上的小时数，一周花在学习上的小时数，注意每行包括五个学生数据：

```
56 6 2   78 7 4   84 5 5   73 4 0   90 3 4
44 9 0   76 5 1   87 3 3   92 2 7   75 8 3
85 1 6   67 4 2   90 5 5   84 6 5   74 5 2
64 4 1   73 0 5   78 5 2   69 6 1   56 7 1
87 8 4   73 8 3  100 0 6   54 8 0   81 5 4
78 5 2   69 4 1   64 7 1   73 7 3   65 6 2
```

代码为：

```
DATA class;
   INFILE 'c:\MyRawData\Exercise.dat';
   INPUT Score Television Exercise @@;

PROC CORR DATA = class;
   VAR Television Exercise;
   WITH Score;
   TITLE 'Correlations for Test Scores';
   TITLE2 'With Hours of Television and Exercise';
RUN;
```

结果为：

```
                    Correlations for Test Scores                    1
                  With Hours of Television and Exercise

                         The CORR Procedure

                1 'WITH' Variables:  Score
                2 'VAR'  Variables:  Television Exercise


                         Simple Statistics

Variable           N      Mean   Std Dev       Sum   Minimum   Maximum

Score             30   74.6333   12.5848    2239.0   44.0000     100.0
Television        30    5.1000    2.3393     153.0         0    9.0000
Exercise          30    2.8333    1.9491   85.0000         0    7.0000


            ❶ Pearson Correlation Coefficients, N = 30
                 ❷ Prob > |R| under Ho: Rho=0

                      Television          Exercise

Score                 ❶ -0.55390          ❶ 0.79733
                      ❷ 0.0015            ❷ 0.0001
```

报告开始于每个变量的描述统计量，接着列出相关矩阵，包括：相关系数（pearson）、P 值。

8.5 使用 proc reg 做简单的回归分析

REG 过程使用最小二乘法拟合线性回归模型，是 SAS/STAT 产品的一部分。Reg 使用逐步法、前进法、后退法进行自变量的筛选。SAS/STAT 其他的产品可以进行非线性、混合线性、logit 回归。SAS/ETS 产品中有时间序列回归的分析。

Reg 只需两步：用 PROC REG 语句开始，用 MODEL 语句指定分析模型。基本形式为：

```
PROC REG;
    MODEL dependent=independent;
```

Model 语句中，自变量在左边，因变量在右边。

Plot 语句是 reg 过程中许多可选的语句之一。可以用 plot 语句产生数据的散点图。如果安装了 SAS/GRAPH 模块，PROC REG 将使用这个模块来产生散点图。产生散点图语句：

```
PLOT dependent*independent;
```

如果没有 SAS/GRAPH 模块，则需要在 proc reg 语句中使用 LINEPRINTER 选项，以产生 plots。由于没有 SAS/GRAPH 模块不能产生回归线，需要用预测值代替观测值来拟合出线。下面的代码显示了用 reg 过程产生数据的单个散点图和预测值：

```
PROC REG LINEPRINTER;
    MODEL dependent=independent;
    PLOT dependent*independent='symbol'P.*independent='symbol'/
        OVERLAY;
```

Symbol 的值指定 SAS 使用哪种标记来标注数据点，如果不指定，SAS 会直接使用数字。P.是代表预测值的关键词。

有很多中选项可以选择，来绘出回归分析的结果。比如可以绘出残差值、学生化残差、Cook's D influence statistics、置信区间。如果有 SAS/GRAPH 模块，那么有很多方法来高质量的控制输出的外观。

例子 在儿童垒球比赛上，有人说，选手多高，他就能将球击多远。想从统计角度来验证，收集了如下的数据，分别是身高（英尺）、三次击球最长的长度（英尺）。

```
50 110   49 135   48 129   53 150   48 124
50 143   51 126   45 107   53 146   50 154
47 136   52 144   47 124   50 133   50 128
50 118   48 135   47 129   45 126   48 118
45 121   53 142   46 122   47 119   51 134
49 130   46 132   51 144   50 132   50 131
```

下面代码读取数据并做回归：

```
DATA hits;
   INFILE 'c:\MyRawData\Baseball.dat';
   INPUT Height Distance @@;
* Perform regression analysis, plot observed values with regression line;
PROC REG DATA = hits;
   MODEL Distance = Height;
   PLOT Distance * Height;
   TITLE 'Results of Regression Analysis';
RUN;
```

在 model 和 plot 语句中，距离是自变量、高度是因变量。输出结果将在 8.6 讨论

8.6 读取 proc reg 的输出

Reg 的输出有几个部分，方差分析和参数估计通常输出在一页。有些选项语句，比如

plot，在另外的页面中产生。

这部分的输出是由如下 proc reg 语句产生的结果：

```
PROC REG DATA = hits;
   MODEL Distance = Height;
   PLOT Distance * Height;
   TITLE 'Results of Regression Analysis';
RUN;
```

第一部分是方差分析的结果，给出了模型对数据拟合的程度：

Results of Regression Analysis 1

The REG Procedure
Model: MODEL1
Dependent Variable: Distance

Analysis of Variance

Source	❶ DF	Sum of Squares	❷ Mean Square	❸ F Value	❹ Pr > F
Model	1	1365.50831	1365.50831	16.86	0.0003
Error	28	2268.35836	81.01280		
Corrected Total	29	3633.86667			

❺ Root MSE	9.00071	R-Square	0.3758
Dependent Mean	130.73333	❼ Adj R-Sq	0.3535
❻ Coeff Var	6.88479		

参数估计的结果如下：

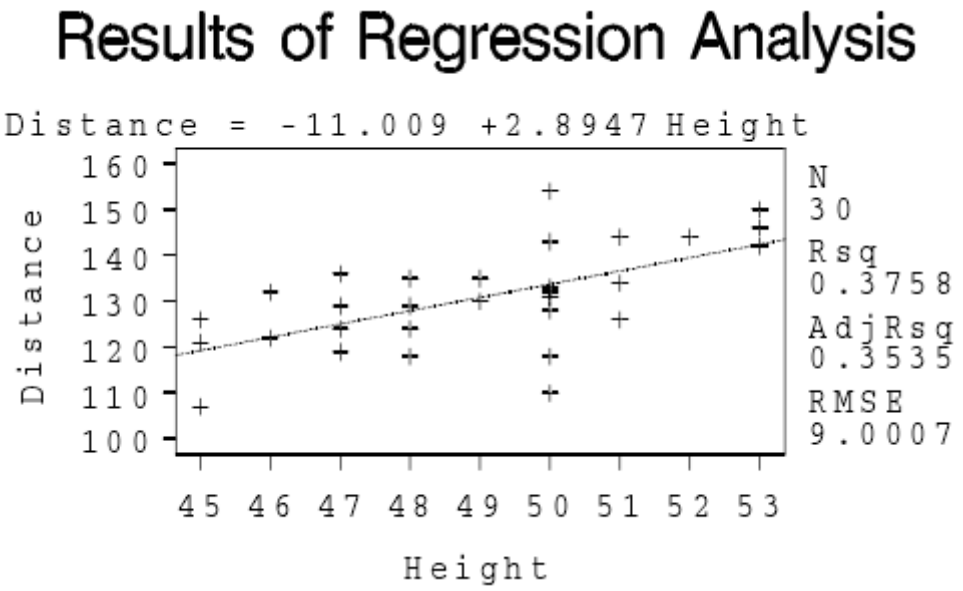
Parameter Estimates

Variable	❶ DF	Parameter Estimate	Standard Error	❷ t Value	❸ Pr > |t|
Intercept	1	-11.00859	34.56363	-0.32	0.7525
Height	1	2.89466	0.70506	4.11	0.0003

参数分析的结果可以构建模型：

Distance=-11.00859+2.89466*Height

下面的图形显示了 plot 语句的结果，如果有 SAS/GRAPH 模块，proc reg 会描出数据点和回归线。上面打印出方程，右边显示统计量：



就这个例子来看，球飞出去的长度确实和击球人的身高有关系，模型是显著的，但是两者之间的关系不是很明显（R-square=0.3758），可能年龄、经验会是比身高更好的预测变量。

8.7 使用 proc anova 做单因素方差分析

Proc anova 是 SAS/STAT 模块的一部分，许可证独立于 BASE SAS。

方差分析有两个基本语句：CLASS 和 MODEL，基本形式为：

```
PROC ANOVA;
    CLASS variable-list;
    MODEL dependent=effects;
```

Class 描述分类变量，并要在 model 语句之前，对于单因素方差分析，只需要列出一个变量。Model 语句描述了自变量和效应（effects）。对于单因素方差分析，效应就是分类变量。每组的观测值数要求一样，这样的数据为平衡的。

Proc anova 有很多选择语句，最常用的是 means，计算出 model 语句中任何一种主效应的自变量的均值。另外，means 还可以做几种多重比较检验，包括 Bonferroni t tests(BON)，Duncan'smultiple-rangetest(DUNCAN)， Scheffe'smultiple-comparison procedure(SCHEFFE)，pairwise t tests(T)，和 Tukey's studentized range test(TUKEY)。Means 语句的基本形式为：

```
MEANS effects/options;
```

Effect 可以为 model 语句中的主效应。选项为希望使用的多重比较检验的名字。

例子 有一份关于几个篮球队员身高的数据，变量为队名、身高，注意每行有六个观测值：

```
red  55 red  48 red  53 red  47 red  51 red  43
red  45 red  46 red  55 red  54 red  45 red  52
blue 46 blue 56 blue 48 blue 47 blue 54 blue 52
blue 49 blue 51 blue 45 blue 48 blue 55 blue 47
gray 55 gray 45 gray 47 gray 56 gray 49 gray 53
gray 48 gray 53 gray 51 gray 52 gray 48 gray 47
pink 53 pink 53 pink 58 pink 56 pink 50 pink 55
pink 59 pink 57 pink 49 pink 55 pink 56 pink 57
gold 53 gold 55 gold 48 gold 45 gold 47 gold 56
gold 55 gold 46 gold 47 gold 53 gold 51 gold 50
```

因为每组都有 12 个队员，所以数据是平衡的。现在想知道哪一组最高，因此还要用 means 语句，并选择 Scheffe's multiple-comparison 过程来比较均值。代码为：

```
DATA basket;
   INFILE 'c:\MyRawData\Basketball.dat';
   INPUT Team $ Height @@;
* Use ANOVA to run one-way analysis of variance;
PROC ANOVA DATA = basket;
   CLASS Team;
   MODEL Height = Team;
   MEANS Team / SCHEFFE;
   TITLE "Girls' Heights on Basketball Teams";
RUN;
```

结果将在 8.8 中讨论：

8.8 读取 proc anova 的输出

Proc anova 的输出至少有两个部分，首先打印出有一个表，给出分类变量的信息：水平数、变量值、观测值数。再次打印出变量表的分析。如果使用了类似 means 这样的语句，那么其结果将接在后面。

8.7 中想要检验是否组与组之间的升高有区别，使用 proc anova 语句如下：

```
PROC ANOVA DATA = basket;
   CLASS Team;
   MODEL Height = Team;
   MEANS Team / SCHEFFE;
   TITLE "Girls' Heights on Basketball Teams";
RUN;
```

第一部分给出了分类变量的信息：

```
                  Girls' Heights on Basketball Teams                1

                        The ANOVA Procedure

                    Class Level Information

          Class         Levels    Values

          Team               5    blue gold gray pink red

                    Number of observations    60
```

分类变量 team 有五个水平：blue,gold,gray,pink,和 red。第二部分是方差分析的表：

Girls' Heights on Basketball Teams 2

The ANOVA Procedure

Dependent Variable: Height

❶ Source	❷ DF	❸ Sum of Squares	❹ Mean Square	❺ F Value	❻ Pr > F
Model	4	228.0000000	57.0000000	4.14	0.0053
Error	55	758.0000000	13.7818182		
Corrected Total	59	986.0000000			

❼ R-Square	❽ Coeff Var	❾ Root MSE	❿ Height Mean
0.231237	7.279190	3.712387	51.00000

Source	DF	Anova SS	Mean Square	F Value	Pr > F
Team	4	228.0000000	57.0000000	4.14	0.0053

因为模型是显著的，因此可以认为不是所有组的队员身高都相同。Means 语句中的 SCHEFFE 选项比较了不同组的身高。

Girls' Heights on Basketball Teams 3

The ANOVA Procedure

Scheffe's Test for Height

NOTE: This test controls the type I experimentwise error rate.

Alpha	0.05
Error Degrees of Freedom	55
Error Mean Square	13.78182
Critical Value of F	2.53969
Minimum Significant Difference	4.8306

Means with the same letter are not significantly different.

Scheffe Grouping		Mean	N	Team
	A	54.833	12	pink
	A			
B	A	50.500	12	gold
B	A			
B	A	50.333	12	gray
B				
B		49.833	12	blue
B				
B		49.500	12	red

8.9 统计分析的图形界面

统计分析的结果也可以用 Graphical User Interfaces 来完成。
SAS Enterprise Guide、分析家 、SAS/LAB 和 SAS/INSIGHT

第九章 导出数据

9.1 导出数据的方法

将数据导出到其他应用程序 有三个基本方法将数据导出到其他应用程序：delimited files 或文本这样其他软件都可以读取的文件。创建一个如 html、rtf、或 xml 这样其他软件业可以读的文件。用其他软件的格式创建数据。

- 不论操作环境是什么，几乎都可以创建分隔文件（delimited files），且大部分软件都是可以读取的。数据步，可以让你很好的控制文件的格式，但是步骤比较多。导出向导（Export Wizard）和导出过程（EXPORT procedure），很好使用，但对结果的控制不是很好。ODS 可以从任何输出中创建逗号分隔的文件（CSV）。
- 使用 ODS，可以从任何输出中创建 HTML、RTF、和 XML 文件，大部分软件都是可以读取这些文件的。
- 如果 PC 文件格式软件中有 SAS/ACCESS 模块，可以创建一些不同的文件类型，这些文件在 PC 程序中很常见。出向导（Export Wizard）和导出过程（EXPORT procedure）都可以产生 PC 文件。通过使用这些程序本身来创建，避免了创建完还要导入。如果没有 SAS/ACCESS 模块，且使用 window 系统，那么可以使用 Dynamic Data Exchange(DDE)或 Open Database Connectivity(ODBC)来讲数据从 SAS 移动到 PC 程序中，且不需要创建中介的文件。

导出 SAS 数据集到其他的操作系统中 有三个有效的方法：交叉环境数据访问（CEDA），XPORT 引擎或 CPORT 过程，XML 引擎，以及 SAS/CONNECT 模块。

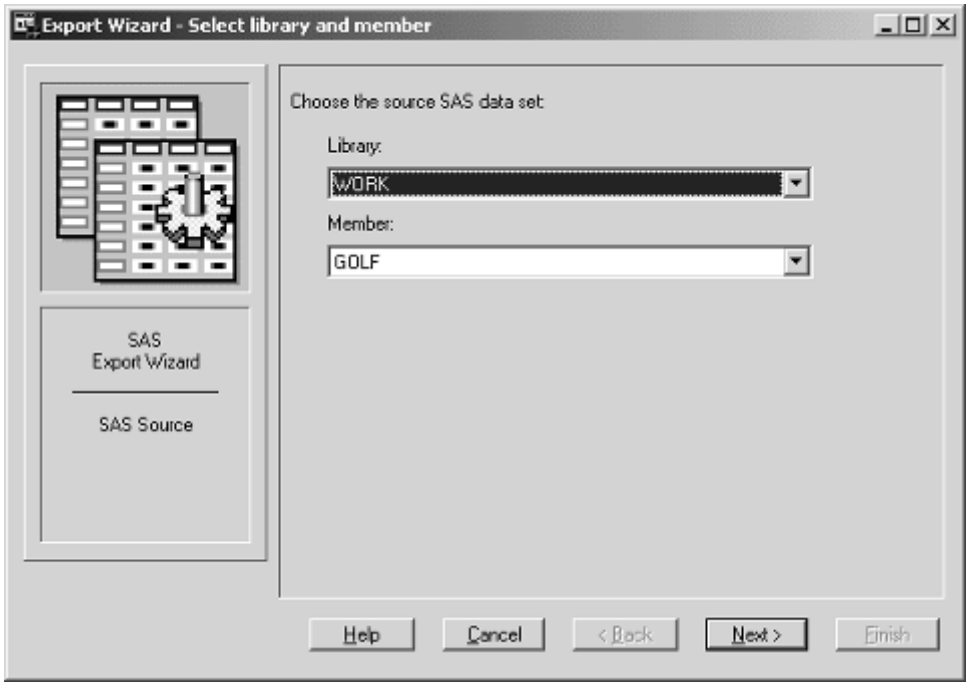
- CEDA 是迄今为止最简单的，将 SAS 数据集移动到其他操作系统的方法。但 CEDA 不能被 SAS version 6 使用，也不能在 OS/390 或 z/OS 中使用。
- XPORT 引擎和 CPORT 过程都创建了可以移动到 其他操作系统的传送文件，再将其转回成 SAS 数据集。创建传送文件会导致数据精度的损失。但对于 SAS version 和 OS/390 或 z/OS，由于不能使用 CEDA，故可能需要这个方法。
- 如果使用的是 SAS 9.0 或更高版本，那么你可以在 LIBNAME 语句中使用 XML 引擎，来创建 XML 文件。这个文件可以转移到其他电脑中，并且可以使用 XML 引擎来转回到 SAS 数据集。
- SAS/CONNECT 模块，连同一些其他的功能，可以让你将 SAS 数据集转移到其他操作系统中，而不需要创建中间文件。SAS/CONNECT 模块也可以讲 SAS 数据集从一个较早版本移到一个较高版本，且反之也可以。

9.2 用导出向导写文件

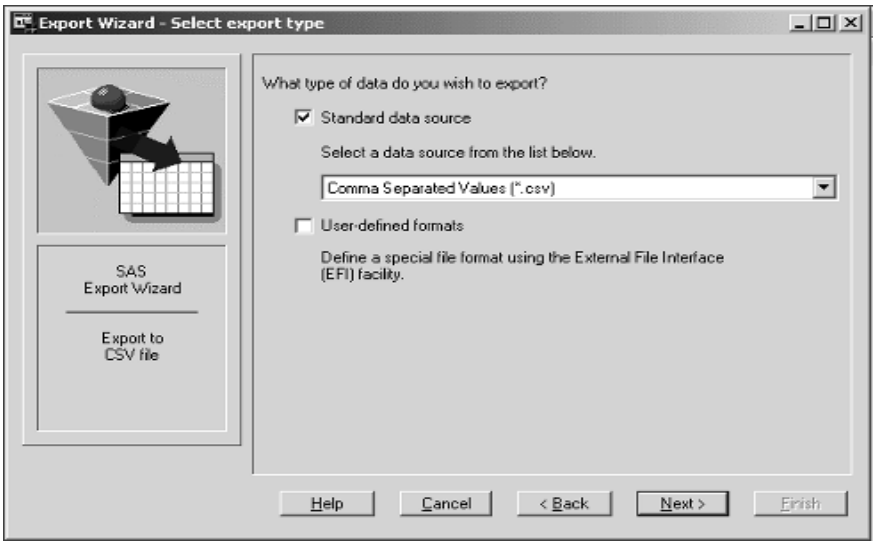
导出向导是一个图形使用界面（GUI），如果你只需要一次性导出数据，那么这个方法比 proc export 简单。

在文件菜单中选择“导出数据…”，在第一个窗口中，选择要导出的 library 和 menber

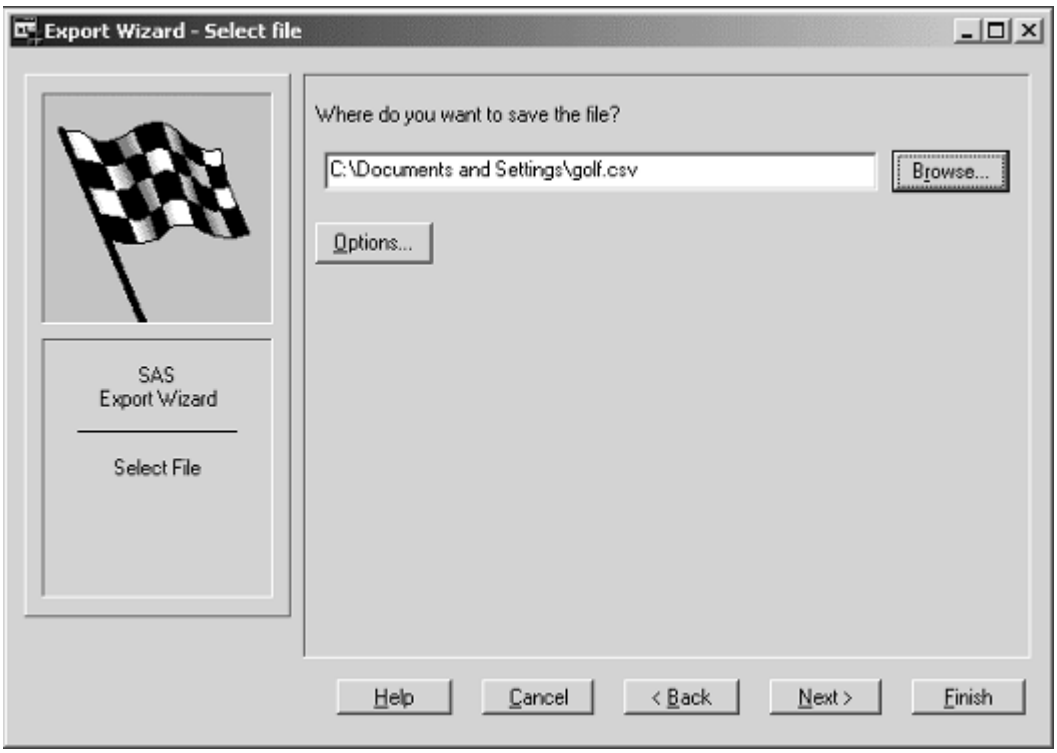
名。如果要导出一个临时 SAS 数据集，那么 library 就是 work。Member 就是 SAS 数据集的名字。



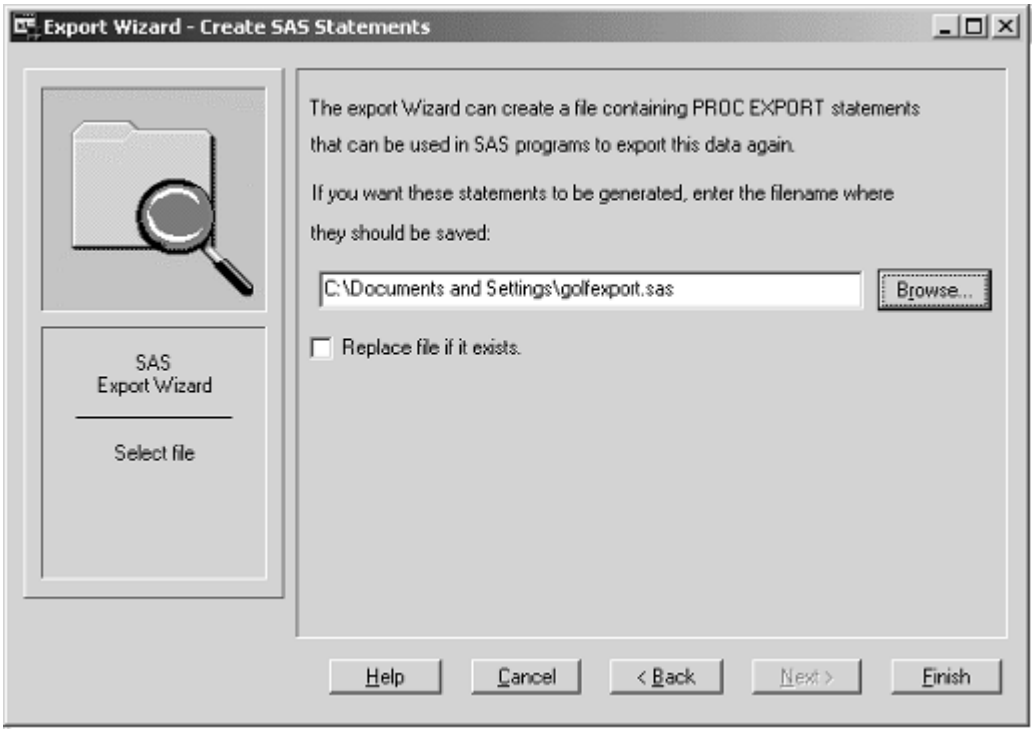
第二个窗口中，选择要创建的文件类型。这可以从 standard data sources 的下拉列表中选择，或勾选 User-defined formats 旁边的框框。



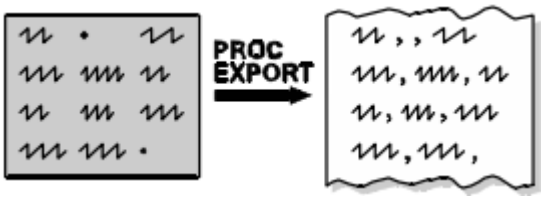
下一个窗口中 ，选择导出文件的路径。如果要导出分割的文件，那么要在 option 中进行设置。



最后一个窗口中，用来储存有导出向导产生的 proc export 语句：



9.3 用 EXPORT Procedure 写分割的文件



使用 EXPORT Procedure 的优点于，它可以将代码融合到现有的 SAS 程序中，每一次创建文件时，不需要通过 export Wizard。

The EXPORT procedure 基本形式为：

PROC EXPORT DATA=data-set OUTFILE='filename';

Data-set 是要导出的数据集，filename 是路径及名称，如下的代码告诉 SAS 读取名为 HOTELS 的临时 SAS 数据集，并写入名为 HOTELs.csv 的逗号分隔的文件中，路径为 C 盘下的 MyRawData 文件夹：

PROC EXPORT DATA=hotels OUTFILE='c:\MyRawData\Hotels.csv';

SAS 根据文件的扩展名，来决定创建哪一种文件。也可以通过在 PROC EXPORT 语句中增加 DBMS=option 来指定文件类型。下面的表格显示了扩展名和对应的 DBMS 辨认器的辨认：

Type of file	Extension	DBMS Identifier
Comma-delimited	.csv	CSV
Tab-delimited	.txt	TAB
Space-delimited		DLM

注意空格分割的文件，是没有扩展名的。因此必须使用 DBMS=option。下面的代码通过 DBMS=option，告诉 SAS 创建一个空格分割的文件，名为 Hotels.spc，替代选项告诉 SAS 替代同名文件。

PROC EXPORT DATA=hotels OUTFILE='c:\MyRawData\Hotels.spc'

DBMS=DLM REPLACE;

如果想创建一个有分隔符的文件，不是逗号、tab 或者空格分隔的，就需要 DELIMITER 语句。DELIMITER 语句不在乎使用什么扩展名，也不在乎指定的哪种 DBMS identifier，文件将会使用你在 identifier 中指定的分隔符。比如，下面的代码产生文件，Hotels.txt，用&作为分隔符：

PROC EXPORT DATA=hotels OUTFILE='c:\MyRawData\Hotels.txt'

DBMS=DLM REPLACE;

DELIMITER='&';

例子 有一份关于高尔夫课程的数据，变量为课程名、洞口数、par、yardage 和 greens fees.

```
Kapalua Plantation 18 73 7263 125.00
Pukalani           18 72 6945  55.00
Sandlewood         18 72 6469  35.00
Silversword        18 71    .  57.00
Waiehu Municipal   18 72 6330  25.00
Grand Waikapa      18 72 6122 200.00
```

下面的代码使用 infile 和 input 语句读取数据并放入名为 GOLF 的永久 SAS 数据集中，存于 C 盘的 MySASLib 路径下。这个例子使用 LIBNAME 告诉 SAS 永久数据集储存在哪里，但也可是使用直接指代：

```
LIBNAME travel 'c:\MySASLib';
DATA travel.golf;
   INFILE 'c:\MyRawData\Golf.dat';
   INPUT CourseName $18. NumberOfHoles Par Yardage GreenFees;
RUN;
```

下面的代码写一个纯文本、tab-delimited 文件，可以用任何文本编辑器来读取：

```
LIBNAME sports 'c:\MySASLib';
* Create Tab-delimited file;
PROC EXPORT DATA = sports.golf OUTFILE = 'c:\MyRawData\Golf.txt' REPLACE;
RUN;
```

由于输出文件以 txt 结尾，且没有 DELIMITER 语句，SAS 将写入一个 tab-delimited 文件，如果运行这段代码，日志将出现如下：

```
NOTE: 7 records were written to the file 'c:\MyRawData\Golf.txt'.
```

注意原始数据中只有六个观测值，多出的一行是变量名。如果用文字编辑器读取这个文件，那么将出现如下：

CourseName	NumberOfHoles	Par	Yardage	GreenFees
Kapalua Plantation	18	73	7263	125
Pukalani	18	72	6945	55
Sandlewood	18	72	6469	35
Silversword	18	71		57
Waiehu Municipal	18	72	6330	25
Grand Waikapa	18	72	6122	200

如果想要改变格式，运行 proc export 之前，在数据步中使用 FORMAT 语句。

9.4 用 EXPORT Procedure 写 PC 文件

如果使用的是 windows 或 unix 系统，且有 SAS/ACCESS 的 PC 文件格式软件，那么就可以使用 EXPORT procedure 来创建 PC 文件类型。如果用的是 Windows 系统，EXPORT procedure 可以创建 Microsoft Access,Microsoft Excel,dBase,和 Lotus files。Unix 系统则可以用 SAS9.1 或更高版本创建 dBase files，也可以创建 Microsoft Access 和 Microsoft Excel files。

Microsoft Excel,Lotus and dBase files PROC EXPORT 创建 Microsoft Excel,Lotus 和 dBase 的基本形式为：

```
PROC EXPORT DATA=data-set OUTFILE='filename';
```

Data-set 是导出的数据集，filename 是输出文件的路径。下面的语句告诉 SAS 读取名为 HOTELS 的临时 SAS 数据集，并写入名为 Hotels.xls 的 Microsoft Excel 文件:

```
PROC EXPORT DATA=hotels OUTFILE='c:\MyRawData\Hotels.xls';
```

SAS 通过扩展名来决定创建哪一种文件。可以通过 DBMS=option 来指定文件类型。下面的表显示了文件扩展名和 DBMS 标识符。

Type of file	Extension	DBMS Identifier
Microsoft Excel	.xls	EXCEL[2]
		EXCEL5
		EXCEL4
Lotus	.wk4	WK4
	.wk3	WK3
	.wk1	WK1
dBase	.dbf	DBF

下面的语句包括 DBMS=option，告诉 SAS 创建一个名为 hotels.xls 的 Microsoft Excel 5 文件，REPLACE 选项意味着替换同名文件。

```
PROC EXPORT DATA=hotels OUTFILE='c:\MyRawData\Hotels.xls'
```

DBMS=EXCEL5 REPLACE;

默认情况下，Microsoft Excel sheet 的名字与 SAS 数据集一样，通过 SHEET=语句可以指定不同的名字（该语句对 Microsoft Excel 4 or Microsoft Excel 5 无效）。Sheet 名中特殊的字符将转化为下划线，且$不允许放在 sheet 名字的后面。下面的代码创建了一个名为 Golf_Hotels 的工作薄：SHEET='Golf Hotels';

Microsoft Access files 创建 Microsoft Access 文件使用 OUTTABLE=option，而不是 OUTFILE=option，并且要增加 DATABASE=语句。基本形式为：

PROC EXPORT DATA=data-set OUTTABLE='filename'DBMS=identifier;
DATABASE='filename';

DATABASE 语句指定了哪一个 Microsoft Access database 文件希望修改或创建，OUTTABLE 选项指定那个 database 的表名。必须要指定 DBMS 选项来创建 Microsoft Access table，下表显示了 DBMS 的标识符：

Type of file	Extension	DBMS Identifier
Microsoft Access	.mdb	ACCESS[3] ACCESS97

例子 仍然是高尔夫课程的数据：

```
Kapalua Plantation 18 73 7263 125.00
Pukalani           18 72 6945  55.00
Sandlewood         18 72 6469  35.00
Silversword        18 71    .  57.00
Waiehu Municipal   18 72 6330  25.00
Grand Waikapa      18 72 6122 200.00
```

下面的代码使用 INFILE 和 INPUT 语句读取数据，并将其放入名为 GOLF 的永久数据集中：

```
LIBNAME travel 'c:\MySASLib';
DATA travel.golf;
   INFILE 'c:\MyRawData\Golf.dat';
   INPUT CourseName $18. NumberOfHoles Par Yardage GreenFees;
RUN;
```

如下代码将 golfSAS 数据集写入 Microsoft Excel 文件：

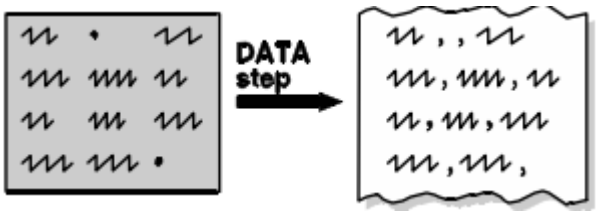
```
LIBNAME sports 'c:\MySASLib;
* Create Microsoft Excel file';
PROC EXPORT DATA=sports.golf OUTFILE = 'c:\MyExcel\Golf.xls' REPLACE;
RUN;
```

Excel 的结果为如下图所示：

	A	B	C	D	E
1	CourseNa	NumberOf	Par	Yardage	GreenFees
2	Kapalua P	18	73	7263	125
3	Pukalani	18	72	6945	55
4	Sandlewo	18	72	6469	35
5	Silverswor	18	71		57
6	Waiehu M	18	72	6330	25
7	Grand Wa	18	72	6122	200

GOLF

9.5 用数据步写原始文件



数据步可以创建原始文件，在数据步中使用 file 语句和 put 语句，可以写下任何一种形式的原始数据文件。虽然 PROC EXPORT 和 Export Wizard 在某种程度上更简单，但选项有限。而数据步可以更灵活的创建。

可以用读取原始数据的同样方式写入原始数据，只需几步改变——用 file 语句和 put 语句。也就是说 infile 和 input 语句是将原始数据文件导入 SAS，而 file 和 put 语句是将原始数据导出。

与 input 一样，put 语句可以用 list,column 或 formatted 风格。但由于 SAS 已经知道变量是否是数值或字符串，因此不需要在字符串变量后加$符号。如果使用 list format 风格，SAS 会自动在变量之间加空格，创建一个空格分隔的文件。要用其他分隔符，在 file 语句中使用 list-style put 语句和 DSD 和 DLM=options：

FILE 'file-specification' DSD DLM='delimiter';

如果使用 column 或 formatted 风格的 put 语句，SAS 将会把变量放在任何你指定的地方。可以用 input 语句中同样的指示器，用@n 移动到第 n 列，用+n 移动 n 列，用/移动到下一行，用#n 滑动到第 n 行，@则停留在当前行，也可以通过引号来加入字符串。

例子 仍然是高尔夫球课程的数据，

```
Kapalua Plantation 18 73 7263 125.00
Pukalani           18 72 6945  55.00
Sandlewood         18 72 6469  35.00
Silversword        18 71    .  57.00
Waiehu Municipal   18 72 6330  25.00
Grand Waikapa      18 72 6122 200.00
```

下面的代码是读取数据的：

```
LIBNAME travel 'c:\MySASLib';
DATA travel.golf;
   INFILE 'c:\MyRawData\Golf.dat';
   INPUT CourseName $18. NumberOfHoles Par Yardage GreenFees;
RUN;
```

现在要将数据写入原始原件，但只有三个变量，按新的变量顺序排列，并且美元符号放入变量 GreenFees 后面。下面的代码读取数据并用 FILE 和 PUT 语句写入原始文件：

```
LIBNAME activity 'c:\MySASLib';
DATA _NULL_;
   SET activity.golf;
   FILE 'c:\MyRawData\Newfile.dat';
   PUT CourseName 'Golf Course' @32 GreenFees DOLLAR7.2 @40 'Par ' Par;
RUN;
```

_NULL_是告诉 SAS 不要新建数据集，以节省电脑资源。

SET 语句告诉 SAS 读取永久数据集 GOLF，FILE 语句告诉 SAS 要创建的输出变量的名字，PUT 语句告诉 SAS 写入的内容和路径。Put 语句包含了两个引用的字符串，“Golf Course”和“Par”，这两个将插入原始文件中。Put 语句使用@列指示符告诉 SAS 变量的变量值放置在哪里。为 GreenFees 变量赋 DOLLAR7.2 格式。用 put 语句可以完全控制原始文件。

运行这段代码，日志窗口将出现下面的信息：

```
NOTE: 6 records were written to the file 'c:\MyRawData\Newfile.dat'.
```

输出结果为：

```
Kapalua Plantation Golf Course $125.00 Par 73
Pukalani Golf Course            $55.00 Par 72
Sandlewood Golf Course          $35.00 Par 72
Silversword Golf Course         $57.00 Par 71
Waiehu Municipal Golf Course    $25.00 Par 72
Grand Waikapa Golf Course      $200.00 Par 72
```

9.6 用 ODS 写分隔和 HTML 文件

ODS 是创造各种输出格式的强有力的工具。在各种输出格式中，ODS 可以创建的有两种，CSV 和 HTML，使用它来将数据从 SAS 中传送到其他程序中，是很有用的方法。许多程序可以读取 CSV 或 HTML 格式，而且任何操作系统都可以使用这种方法。

由于所有的过程其输出都经过 ODS，因此可以通过选择正确的目的地来用 ODS 导出数据，并用 proc print 来获取数据列表。如果不想让 SAS 打印缺失数据，可以使用 MISSING='' 系统选项。默认下，PROC PRINT 打印观测序号，可以通过使用 NOOBS 让其不打印。

CSV files 从 9.0 开始，可以用 ODS 创建 CSV 文件。CSV 文件用逗号将变量值分开，且变量值被括在双引号之内。双引号允许变量值包含逗号。创建一个 CSV 文件来转载数据，用下面的语句：

```
ODS CSV FILE='filename.csv';
    Your PROC PRINT statements go here
RUN;
ODS CSV CLOSE;
```

filename.csv 是要创建的 CSV 文件，并且下面要为你的数据插入正确的 proc print 语句。如果想要在 CSV 文件中包含标题和注脚，使用 csvall，而不是 csv。

HTML files 使用下面语句为你的数据产生 HTML 文件，可以在 ODS HTML 语句中增加 STYLE=option 语句来选择不同的风格。或者，如果不想使用任何风格，则使用 CHTML 输出目的地来代替 HTML。

```
ODS HTML FILE='filename.html';
    Your PROC PRINT statements go here
RUN;
ODS HTML CLOSE;
```

例子 此例使用 SAS 永久数据集 GOLF，仅包括在 hawaii 的 gofl 课程。下面的代码使用 ODS 来创建 CSV 文件，golfinfo.csv：

```
LIBNAME travel 'c:\MySASLib';
ODS CSV FILE='c:\MyCSVFiles\golfinfo.csv';
PROC PRINT DATA = travel.golf;
  TITLE 'Golf Course Information';
RUN;
ODS CSV CLOSE;
```

CSV 文件如果用记事本打开，如下所示：

```
"Obs","CourseName","NumberOfHoles","Par","Yardage","GreenFees"
"1","Kapalua Plantation","18","73","7263","125"
"2","Pukalani","18","72","6945"," 55"
"3","Sandlewood","18","72","6469"," 35"
"4","Silversword","18","71","    .","  57"
"5","Waiehu Municipal","18","72","6330"," 25"
"6","Grand Waikapa","18","72","6122","200"
```

如果使用 excel 打开，则如下所示:

	A	B	C	D	E	F
1	Obs	CourseNar	NumberOf	Par	Yardage	GreenFees
2	1	Kapalua P	18	73	7263	125
3	2	Pukalani	18	72	6945	55
4	3	Sandlewoc	18	72	6469	35
5	4	Silversworc	18	71	.	57
6	5	Waiehu Mu	18	72	6330	25
7	6	Grand Wai	18	72	6122	200
8						

golfinfo

下面的代码从 golf 数据创建一个 HTML 文件，golfinfo.html，并且在 proc print 语句中使用 noobs 选项来减少观测之列：

```
LIBNAME travel 'c:\MySASLib';
ODS HTML FILE='c:\MyHTMLFiles\golfinfo.html';
PROC PRINT DATA = travel.golf NOOBS;
   TITLE 'Golf Course Information';
RUN;
ODS HTML CLOSE;
```

用 excel 打开：

Golf Course Information

CourseName	NumberOfHoles	Par	Yardage	GreenFees
Kapalua Plantation	18	73	7263	125
Pukalani	18	72	6945	55
Sandlewood	18	72	6469	35
Silversword	18	71	.	57
Waiehu Municipal	18	72	6330	25
Grand Waikapa	18	72	6122	200

golfinfo

9.7 和其他类型电脑分享 SAS 数据集

访问 SAS 数据集时候，SAS 会自动检测数据以决定是否和与你使用的操作系统兼容。如果数据集处于不同的系统之中，SAS 会自动使用交叉环境数据访问（CEDA），以动态传送数据至一种形式，这种形式下，SAS 可以在你的系统中理解数据。有两种 CEDA 不能使用的情况，分别为：OS/390 或 z/OS，和 SAS 6.0 及以前的版本。

决定数据表现 CEDA 是透明的，以至于你不知道什么时候它在被使用。如果想知道什么时候 CEDA 被使用，可以用下面的语句：OPTIONS MSGLEVEL=I;

则，无论什么时候 SAS 使用 CEDA 访问数据，日志窗口都会出现下面的类似信息：

```
INFO: Data set TEST.MYUNIX.DATA is in a foreign host format. Cross Environment
Data Access will be used, which may require additional CPU resources and reduce
performance.
```

为外部主机创建 SAS 数据集 创立一个 SAS 数据集，方便其他电脑系统访问，而又不需要浪费电脑资源转换格式，可以在 libname 语句中使用 OUTREP=option，基本形式：

LIBNAME libref'path'OUTREP=data-representation;

如果想要那个库中所有数据集都有指定 host representation，还可以：

data-set-name(OUTREP=data-representation)

data-representation 基本来说是操作系统的名字，比如想要 data representation 是 Microsoft Windows 64-Bit Edition，那么就应该为 WINDOWS_64；如果想要 data representation 是 Solaris 32-Bit Edition，那么应该为 SOLARIS_32。

例子 你有关于 HAWII 高尔夫球课程的数据，储存在你的电脑里。可是你的朋友需要你的数据，而他的电脑是 linux 系统，现在需要把数据集转换成 linux 格式。下面的代码在数据步中使用 SET 语句读取 sport 逻辑库中的 golf 数据，Data 语句中的 OUTREP=data set 选项告诉 SAS 用 linux 格式写下数据，并创建名为 golflinux 的数据集。

```
OPTIONS MSGLEVEL=I;
LIBNAME sports 'c:\MySASLib;
DATA sports.golflinux(OUTREP=LINUX);
  SET sports.golf;
RUN;
```

系统选项 MSGLEVEL=I 导致了下面的信息：

```
INFO: Data set SPORTS.GOLFLINUX.DATA is in a foreign host format. Cross
Environment Data Access will be used, which may require additional CPU resources
and reduce performance.
```

移动 SAS 数据集 如果两台电脑都访问同一个文件系统，那么用 LIBNAME 语句指向 SAS 数据集存放的路径就可以了，否则要用 binary 模式的 FTP，或者使用外部媒介，比如软盘或 CD。

FAT 文件系统 从 SAS 版本 7 开始，创建的数据集默认扩展名为.sas7bdat。有的使用 FAT 文件系统的 windows 系统只能有三个字符的扩展名，此时 SAS 数据集的扩展名为.sd7。如果接受三个字符的扩展名，创建一个三个字符扩展名的数据集，则需要再 libname 语句中使用 SHORTFILEEXT 选项：

```
LIBNNAME libref'path' SHORTFILEEXT;
```

如果你不能使用 CEDA