统计软件教程

李东风 编著

北京大学数学科学学院
2002年3月

前言

统计学是研究如何收集数据、分析数据并进行推断的学科。统计学的应用必然要涉及数据的收集、存贮、整理，以及各种统计方法的实现，这些都要靠统计软件的帮助来完成。

从电子计算机出现至今，统计软件已经有了长足的发展。一方面经典的统计方法都已被实现到统计软件中，另一方面，帮助统计学家实现新的统计方法的软件也极大地推动了新的统计计算方法的研究与开发。国内统计软件的应用这些年来也有很大进步，引进了国外的SAS, SPSS, Splus, Gauss等统计软件并出版了一些有关书籍。然而，现在介绍统计软件的书多数是着重讲软件本身的功能，没有考虑到初学者循序渐进的要求，可供自学的教材性质的书比较少，所以本书试图填补这一空白，以自学教材的形式讲述，不强调全面介绍软件功能，而是让读者能够通过阅读本书掌握统计软件的基本用法，然后可以进一步阅读更详细的资料以达到熟练使用统计软件完成统计应用的目的。

本书主要讲SAS系统和Splus系统。SAS系统是一个集大型数据库管理、统计分析、报表图形、信息系统开发等多种强大功能为一体的大型软件系统，是国际公认的优秀统计分析软件，但初学有些困难，本书力图采取循序渐进的讲法使读者轻易掌握SAS的要点。Splus是一种面向对象的统计编程与统计分析系统，它具有强大的探索性数据分析功能，实现了很多最新的统计方法，而且用Splus编制

统计计算程序既容易学会又容易使用，可以极大地降低统计计算程序的开发时间，方便了统计学家研究新的统计方法和进行统计计算。另外，有一种自由软件R和Splus基本兼容，在我们现在保护知识产权的情况下R是一个不错的统计软件选择。

本书可用作大学统计软件课程的教材或用作自学教程。阅读本书需要读者有基本的数理统计知识，另外，读者最好已经学过计算机程序设计，本书讲的是专用语言，不适合作为一般的程序设计入门教材。本书的着重点是用统计软件进行数据管理、统计分析和统计计算编程，适用于统计软件的初学者，对于用统计软件开发商业化应用程序所需知识涉及较少，希望进行统计应用开发的读者可以在阅读本书后进一步学习有关知识。

目录

第一章 SAS初阶

1.1 初识SAS

1.1.1 启动

用如下方法可以进入SAS系统的窗口运行环境：

在Windows环境中, 从开始菜单的程序文件夹中找到SAS系统文件夹, 从中启动SAS系统。或者生成SAS.EXE的快捷方式（把SAS.EXE用鼠标右键拖到桌面）, 双击SAS.EXE启动。

1.1.2 SAS AWS（SAS应用工作空间）

启动后, 出现如图1.1的SAS运行界面, 术语称为“SAS工作空间（SAS Application WorkSpace）”。这是SAS V8.1的界面。它象其它Windows应用程序一样, 在一个主窗口内, 包含若干个子窗口, 并有菜单条、工具栏、状态栏等。

SAS有三个最重要的子窗口：程序窗口（PROGRAM EDITOR）、运行记录窗口（LOG）、输出窗口（OUTPUT）。

程序窗口的使用类似于Windows中的记事本程序, 可以在其中编辑文本文件, 主要是编辑SAS程序。SAS V8.1的程序编辑功能有所增强, 现在可以用

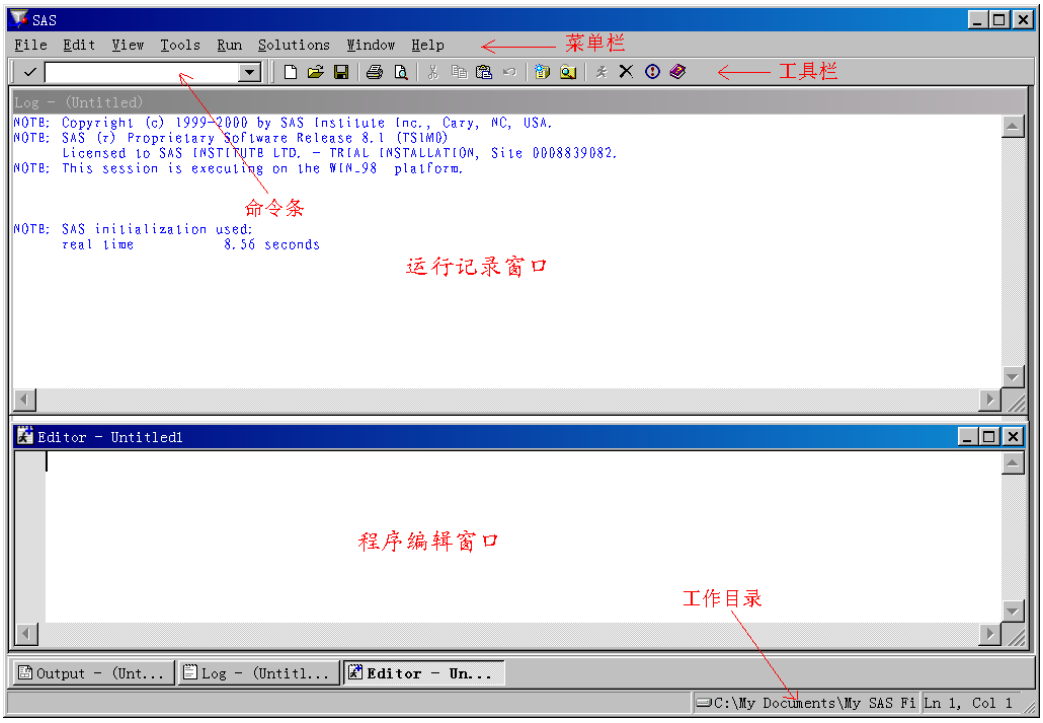


图 1.1: SAS AWS

不同颜色显示不同的SAS程序部分, 可以自动缩进排列程序文本, 可以折叠一段程序。程序可以直接在窗口中键入, 插入新行用回车, 插入点光标（闪动的竖线）可以用光标键（上下左右箭头、Home、End）移动或用鼠标单击到某一处。按住Shift再按光标键可以加亮显示一块文本, 然后用复制、剪切、粘贴命令（Edit菜单中的Cut、Copy、Paste, 或工具栏图标）可以复制或移动加亮显示的文本。这些编辑操作具体请参考Windows的有关文档。

运行记录窗口记录程序的运行情况, 运行是成功还是出错, 运行所用时间, 如果出错, 错在什么地方。运行记录窗口中以红色显示的是错误信息。

输出窗口显示SAS程序的文本型输出（图形输出单独有一个GRAPHICS 窗口）。输出分页显示。

要把光标移动到某一窗口, 可以用主菜单中的Window菜单选择要显示的窗口。用功能键F5可以切换到程序窗口, F6可以到运行记录窗口, F7可以到输出窗口。

SAS主窗口标题栏下是主菜单。SAS菜单是动态的, 其内容随上下文而不同, 即光标在不同窗口其菜单也不同。其中, File（文件）菜单主要是有关SAS文件调入、保存、转换及打印的功能。Edit（编辑）菜单用于窗口的编辑（如清空、复制、剪切、粘贴、查找、替换）。View菜单可以打开或切换到SAS的各个工作窗口, 如程序编辑窗、SAS管理器。Tools菜单可以打开一些SAS提供的小工具如图象编辑器, 还包含修改SAS运行选项的功能。Run菜单用于程序执行、远程调用等, 只在当前窗口是程序窗时有效。Solutions菜单是SAS的一些图形界面操作的模块的入口, 比如SAS/INSIGHT、ASSIST等。

主菜单下是一个命令条和工具栏菜单。命令条主要是用于与SAS较早版本的兼容性, 可以在这里键入SAS的显示管理命令。工具栏图标提供了常见任务的快捷方式, 比如保存、打印、帮助等等。鼠标光标在某一工具栏图标上停留几秒可以显示一个说明。工具栏也是动态的, 当光标在编辑窗口时工具栏图标的解释如下：

- New — 建立新的编辑窗口。
- Open — 打开文件到编辑窗口。用户指定一个

文件调入到编辑窗口内。这个文件从此与编辑窗口相关联, 以后的存盘操作将自动存入这个文件。

 Save — 存盘, 保存编辑窗口内容, 注意如果此窗口已经与一个文件相联系的话此功能将覆盖文件的原有内容而不提示。

 Print — 打印当前窗口内容

 Print preview — 打印预览。

 Cut — 剪切选定文本。

 Copy — 复制选定文本。

 Paste — 粘贴。注意这些操作是对Windows剪贴板进行的, 可以用来与其它Windows应用程序交换文本、数据等。剪切或复制到剪贴板的内容可以被其它应用程序粘贴, 其它应用程序放到剪贴板的内容也可以粘贴到SAS的编辑窗口中。

 Undo — 撤销刚才的编辑操作。

 New Library — 建立新的SAS库。

 SAS Explorer — 打开SAS管理器窗口查看、管理SAS的各个库和库中的文件。这是较新版本SAS系统的增强功能, 我们可以比较方便地利用这个窗口来管理我们的SAS文件。

 Submit — 提交编辑窗口中的程序

 Clear All — 清空当前窗口内容。

 Break — 中断正在运行的SAS程序。

 Help — 进入SAS的帮助界面。因为SAS是一个十分庞大的系统, 我们不可能完全了解SAS的每一个细节, 所以善于利用SAS的帮助系统对于学好用好SAS是极为重要的。也可以从菜单进入SAS帮助。

SAS界面的状态栏中包含了当前工作目录，这是

文件打开、保存的缺省目录。双击此处可以更改当前工作目录。

1.1.3 简单运行样例

假设我们有一个班学生的数学成绩和语文成绩, 数学满分为100, 语文满分为120, 希望计算学生的平均分数（按百分制）并按此排名, 可以在程序窗口输入此程序：

```
title '95级1班学生成绩排名';
data c9501;
  input name $ 1-10 sex $ math chinese;
  avg = math*0.5 + chinese/120*100*0.5;
  cards;
李明       男 92 98
张红艺     女 89 106
王思明     男 86 90
张聪       男 98 109
刘颍       女 80 110
;
run;
proc print;run;
proc sort data=c9501;
  by descending avg;
run;
proc print;run;
```

比较老的SAS版本(SAS V8.0及更低)对中文输入支持不够好, 要输入这样包含中文的程序最好是打开一个其它的编辑程序如Windows中的记事本, 在记事本中复制输入的程序, 然后到SAS系统程序窗口中使用粘贴命令(用Edit菜单的Paste或工具栏上的粘贴图标), 把程序复制到SAS编辑窗口中。也可以在记

事本或其他程序编辑器中把编好的程序存盘, 然后在SAS程序窗口用File菜单的Open命令打开保存好的程序文件。从SAS V8.1开始SAS对中文输入已经消除了原有的问题所以我们可以直接在SAS程序编辑窗口输入SAS程序。

要运行此程序, 只要用鼠标单击工具栏的提交图标 , 或用Run菜单的Submit命令。运行后, 运行记录窗口出现如下内容：

```
1    title '95级1班学生成绩排名';
2    data c9501;
3      input name $ 1-10 sex $ math chinese;
4      avg = math*0.5 + chinese/120*100*0.5;
5      cards;

NOTE: The data set WORK.C9501 has 5 observations and 5 variables.
NOTE: DATA statement used:
      real time           0.60 seconds


11   ;
12   run;
13   proc print;run;

NOTE: There were 5 observations read from the data set WORK.C9501.
NOTE: PROCEDURE PRINT used:
      real time           0.33 seconds


14   proc sort data=c9501;
15     by descending avg;
16   run;

NOTE: There were 5 observations read from the data set WORK.C9501.
NOTE: The data set WORK.C9501 has 5 observations and 5 variables.
NOTE: PROCEDURE SORT used:
      real time           0.05 seconds


17   proc print;run;

NOTE: There were 5 observations read from the data set WORK.C9501.
NOTE: PROCEDURE PRINT used:
      real time           0.00 seconds
```

其中记录了每段程序的运行情况、所用时间、生成数据保存情况。如果有错误还会用红色指示错误。比如, 最后的`proc print`后面的分号如果丢失,

记录窗口显示如下错误：

```
NOTE: SCL source line.
34   proc printrun;
          --------
          181
ERROR 181-322: Procedure name misspelled.
```

错误说明为过程名错拼, 但实际上是丢了分号导致print和run连成了一个词。在程序窗口修改并正确运行后输出窗口出现如下结果：

95级1班学生成绩排名 4

Obs	name	sex	math	chinese	avg
1	李明	男	92	98	86.8333
2	张红艺	女	89	106	88.6667
3	王思明	男	86	90	80.5000
4	张聪	男	98	109	94.4167
5	刘颖	女	80	110	85.8333

95级1班学生成绩排名 5

Obs	name	sex	math	chinese	avg
1	张聪	男	98	109	94.4167
2	张红艺	女	89	106	88.6667
3	李明	男	92	98	86.8333
4	刘颖	女	80	110	85.8333
5	王思明	男	86	90	80.5000

这里有两页输出, 第一页是输入数据后用PROC PRINT显示的数据集, 第二页为按平均分排名后的结果。

从上面的例子程序可以看出SAS程序的一些特点。SAS程序由**语句**组成, 语句用分号结束。SAS程序中大小写一般不区分(字符串中要区分大小写)。SAS程序中的空格、空行一般可以任意放置, 这样我们可以安排适当的缩进格式使得源程序结构清楚易读。SAS程序由两种“步”构成, 一种叫

数据步(data step), 一种叫**过程步**(proc step), 分别以DATA语句和PROC语句开始。数据步和过程步由若干个语句组成, 一般以RUN语句结束。

1.2 SAS基本概念

本节介绍一些SAS特有的概念, 其中最重要的是数据集。

1.2.1 SAS数据集

SAS数据集（SAS Datasets）可以看作由若干行和若干列组成的表格, 类似于一个矩阵, 但各列可以取不同的类型值, 比如整数值、浮点值、时间值、字符串、货币值等等。SAS数据集存放在以特殊格式存放的二进制文件中, 我们用一个SAS中的逻辑名来使用SAS数据集而不需关心它到底如何存储在磁盘上。比如, 1.1.3的例子生成了一个名为C9501的数据集, 它的逻辑形式如下表：

表 1.1: C9501数据集的逻辑形式

NAME	SEX	MATH	CHINESE	AVG
李明	男	92	98	86.8333
张红艺	女	89	106	88.6667
王思明	男	86	90	80.5000
张聪	男	98	109	94.4167
刘颖	女	80	110	85.8333

数据集的每一行叫做一个观测(Observation), 每列叫做一个变量(Variable)。 SAS数据集等价于关系数

据库系统中的一个表, 实际上一个SAS数据集有时也称作一张表。在数据库术语中一个观测称作一个记录, 一个变量称作一个域。在C9501数据集中有5个观测, 分别代表5个学生的情况, 而每个学生有5个属性值, 分别为姓名、性别、数学成绩、语文成绩、平均分, 所以此数据集有5个变量。

从上面看出, 数据集要有名字, 变量要有名字, SAS中对**名字**(数据集名、变量名、数据库名, 等等)有约定：SAS名字由英文字母、数字、下划线组成, 第一个字符必须是字母或下划线, 名字最多用8个字符, 大写字母和小写字母不区分。比如, name, abc, aBC, x1, year12, _NULL_ 等是合法的名字, 且abc和aBC是同一个名字, 而class-1(不能有减号)、a bit(不能有空格)、serial#(不能有特殊字符)、Documents(超长)等不是合法的名字。

1.2.2 SAS数据库

SAS数据集是各种特殊格式的**SAS文件**中最重要的一种。另一种重要的SAS文件是**SAS目录**(Catalog), 用来保存各种不能表示成行列结构表格形式的数据, 比如系统设置、图象、声音等。多个SAS文件可以放在一起, 称为一个**SAS数据库** (Library)。数据库有一个库名(Libname), 其命名遵循上述SAS名字命名原则。在MS DOS / Windows环境中, 一个SAS数据库实际是磁盘上的一个子目录(特殊情况下一个数据库可以由几个子目录组成)。为了把库名和子目录联系起来, 使用LIBNAME语句。比如, 我们在`C:\Y1995`子目录中保存了几个SAS数据集, 可以用如下语句把

库名MYLIB 与子目录C:\Y1995联系起来：

```
libname mylib "c:\y1995";
```

在SAS图形界面中工作时也可以单击建立新库(即把一个库名如MYLIB 和一个实际的子目录如C:\Y1995 联系起来)。见图1.2。

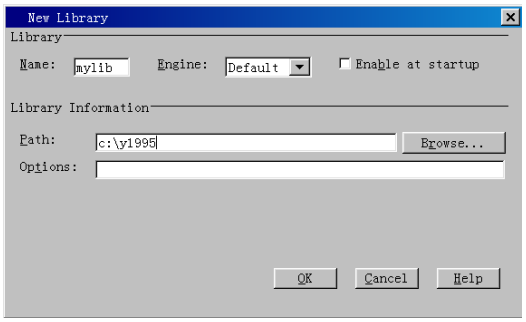


图 1.2: 建立新库

有三个预定义的SAS数据库: WORK, SASUSER, SASHELP。其中, WORK数据库叫做**临时库**, 存放在其中的SAS文件叫**临时文件**, 这些临时文件当退出SAS系统时会被自动删除。SASUSER库保存与用户个人设置有关的文件, 它是永久的, 即退出SAS时文件不会被删除。SASHELP库保存与SAS帮助系统、例子有关的文件, 是永久的。从上面看出, SAS文件分为**临时文件**和**永久文件**：临时文件在退出SAS系统时自动被删除, 永久文件在退出SAS系统时不自动被删除。所以, 我们把作为中间结果使用的数据集或练习用的数据集作为临时数据集保存, 而需要以后再用的数据集则可以保存为永久数据集。

临时数据集和永久数据集的区别是：临时数据集可以用**单水平名**, 即只有数据集名, 比如C9501, 而永久数据集名由两部分组成, 前一部分是它的库名, 后一部分才是数据集名, 两部分中间用小数点连接, 比如放在MYLIB库(即“C:\Y1995”子目录)中的数据集TEACH必须用MYLIB.TEACH表示。这样指定的

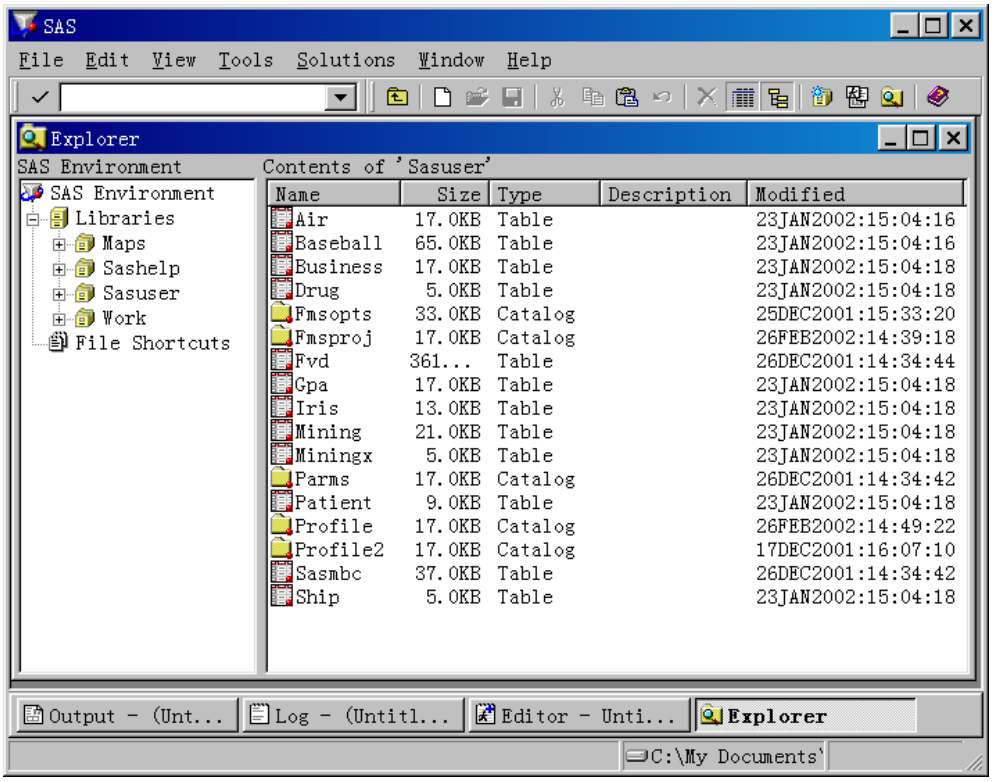


图 1.3: SAS 管理器

数据集名在生成时可以放到由库名指定的子目录中, 在读取时可以到指定的子目录读取, 并且不会被自动删除。

临时数据集除用单水平名外, 也可以用库名为WORK的两水平名, 如WORK. C9501 和C9501 是一样的。

要生成永久数据集, 只要在指定要生成的数据集名时使用两水平名且库名已有定义, 比如, 要把上面的C9501数据集在生成时就放到“C:\Y1995”子目录中, 可以用如下语句:

```
libname mylib "c:\y1995";
data mylib.c9501;
```

```
.....
proc sort data=mylib.c9501;
.....
```

这个程序和1.1.3的例子相比只是增加了一个定义库名的LIBNAME语句, 然后在所有用到数据集名C9501的地方换成了两水平名MYLIB.C9501。要注意生成的数据集是MYLIB.C9501后面在用到它的时候(在PROC SORT 中) 也必须使用两水平名MYLIB.C9501而不能使用单水平名C9501, 这两个名字指向的不是同一个SAS文件。

新版的SAS提供了SAS管理器(SAS Explorer)来管理数据库和里面的文件。刚启动的SAS界面中可能已经打开了管理器窗口, 但是窗口是靠向一边的, 从Window菜单中取消Docked选择可以把它变成一般窗口。如果这个窗口没有打开, 可以从View菜单中选择Explorer把它打开, 或单击工具栏的图标 。图1.3显示了打开管理器后的SAS界面。这个界面分左右两个部分, 左边以树形显示了SAS工作环境的内容, 最主要的是Libraries项即当前有定义的数据库。在左边选择一个库如USER可以显示库中内容的列表。 形的图标表示SAS数据集, 形的图标表示SAS目录(Catalog)。

管理器窗口的工具栏也与程序编辑窗口的工具栏有所不同, 而且相同的图标也可能表示不同操作。比如, 表示建立新的数据集(表)、目录、查询(Query)或新库, 、 和 分别表示SAS文件的剪切、复制、粘贴, 表示删除文件或库。

这里也有几个新的图标。表示树形结构向上一层，表示是否显示库内文件的细节(类型、描述、创建日期等)。控制管理器窗口的树形显示是否打开。返回到程序编辑窗口界面。

双击管理器中的一个SAS文件可以打开此文件进行查看或编辑，如果打开的是SAS数据集，则进入SAS的VIEWTABLE界面，这是一个功能十分强大的数据查看与编辑界面。VIEWTABLE不需要把数据全都读入内存中。一般练习时我们不容易发现VIEWTABLE的优越性，但是当我们用SAS连接一个网络数据库并直接访问其中巨大的表或视图时就会发现其设计的方便合理性。我们后面会看到模块INSIGHT也能直观地浏览数据集内容并进行修改，但是INSIGHT不能处理过于巨大的数据集。

1.3 不需编程的SAS应用— SAS / INSIGHT

1.3.1 介绍

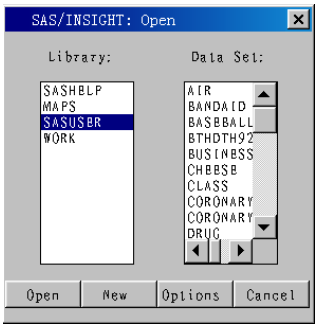


图 1.5: SAS/INSIGHT: 选择数据集

SAS的使用方法一般是象1.1.3那样输入一个程序，运行，修改，最后在输出窗口得到结果。随着图形界面、用户友好等程序思想的发展，SAS也逐渐提供了一些不需要学习SAS编程就能进行数据管理、分析、报表、绘图的功能，其中做得比较出色的一个是SAS/INSIGHT模块。SAS / INSIGHT是在基本的SAS 系统基础上添加的一个模块，提

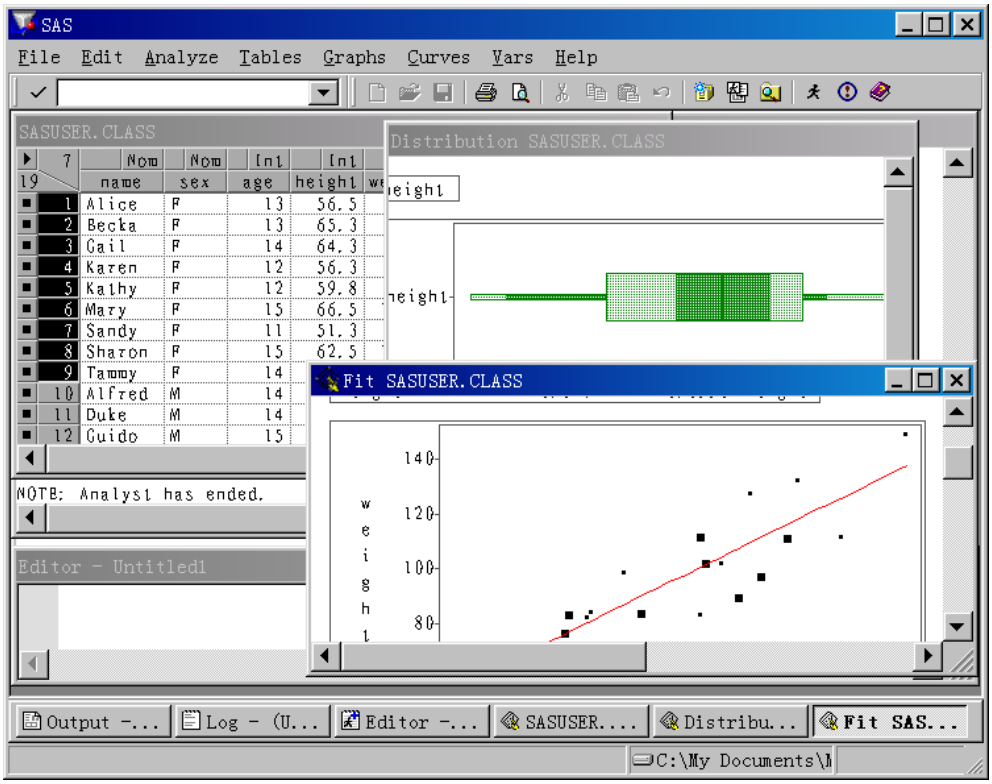


图 1.4: SAS/INSIGHT

供了数据交互输入、数据探索、分布研究、相关分析、各种图形等功能。另外一个类似的模块叫Analyst, 它为常见的数据管理和统计功能提供了一个简单易用的图形界面, 还能够把用图形界面进行的操作以普通SAS程序的形式记录下来供我们学习参考。SAS/INSIGHT 在数据探索方面比较有特色, 这里我们先初步介绍SAS/INSIGHT 的使用, SAS/INSIGHT 的一些统计功能在后面的有关章节介绍, 第三章将介绍Analyst模块。

要启动SAS/INSIGHT, 选“Solutions - Analysis - Interactive Data Analysis”菜单, 首先出现图1.5那样的选

择数据集的窗口, 这是SAS/INSIGHT必须先选择一个要分析、观察的数据集。如果要生成新数据集, 按New按钮, 如果要打开已有数据集, 按Open按钮。图1.4是SAS/INSIGHT运行时的样子。

1.3.2 数据窗口

图 1.6: SAS/INSIGHT数据窗口

SAS/INSIGHT提供了一个类似于电子表格的数据窗口来管理数据集。图1.6为显示了数据集SASUSER.CLASS [1] 的数据窗口, 此数据集是一个班19个

[1]如果没有找到这个数据集, 按如下步骤可以生成此数据集及其他例子数据集: 选“Solutions - Analysis - Analyst”进入Analyst, 然后选其中的“Tools - Sample Data”, 在出现的对话框中选中所有数据集就可以生成例子数据。

学生的一些情况, 包括姓名、性别、年龄、身高、体重。我们看到, 数据窗口标题行显示了打开的数据集的名字, 标题行下左上角位置有一个向右的小三角, 这是数据窗口的菜单, 见图1.7。三角下方和右方分别显示了观测个数和变量个数。窗口内每行最左边的方块是观测的绘图标记, 用于在图形中标记观测；然后是观测序号；再往右是各变量的值。数据窗口中的各变量用作列标题, 如图1.6中的name、sex、age等就是数据集SASUSER.CLASS 中的五个变量的名字。在每一个变量名的上面有两个标签, 右边一个代表变量的量测水平, 分为区间变量（Int）和名义变量（Nom）。区间变量是取连续值的变量, 只能为数值；名义变量是取离散值的变量, 一般为字符型, 也可以取数值。变量名上面左边的标签代表变量在分析中的缺省用途, 比如Label表示变量的值在绘图中用来标记观测, Group表示变量的值用来分组, 等等。

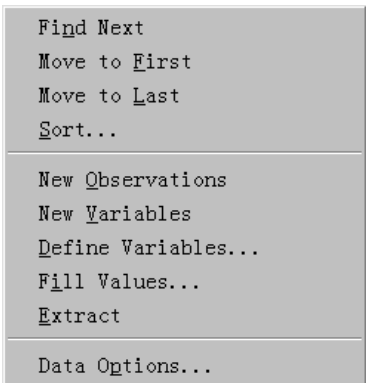


图 1.7: SAS/INSIGHT数据窗口菜单

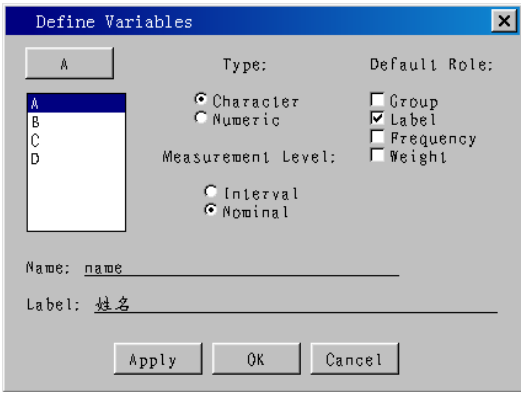


图 1.8: SAS/INSIGHT: 定义变量

数据窗口可以用来建立新数据集。在SAS / IN-

SIGHT内用“File - New” 菜单或在启动INSIGHT的窗口(图1.5)按“New”按钮, 将出现一个空的数据窗口。这时, 可以直接向第一行输入数据, 比如要输入1.1.3中的C9501数据集, 就可以在第一行的前四列中分别输入李明、男、92、98, 这时各列自动取变量名为A, B, C, D, 而且量测水平自动定为前两个字符型是名义变量（Nom）, 后两个数值型是区间变量（Int）。为了修改变量名和变量的用途, 从数据窗口的菜单(图1.7) 选Define Variables, 出现图1.8的定义变量窗口, 在这里可以修改变量名, 给变量加标签(Label), 可以选择变量的量测水平, 可以规定变量的用途。变量的标签是对变量的一个可以长达40个字符的描述, 可以用于以后的输出, 可以用汉字。

定好变量名等属性后就可以继续输入其它数据行, 每输入一行后回车, 直至把全部数据输完。为了使回车时光标从前一行尾部进到下一行第一格, 可以从数据窗口菜单(图1.7) 中选“Data Options”, 在弹出的对话框中(图1.9) 选择回车的方向(Direction of Enter)为左下(Down and Left)。

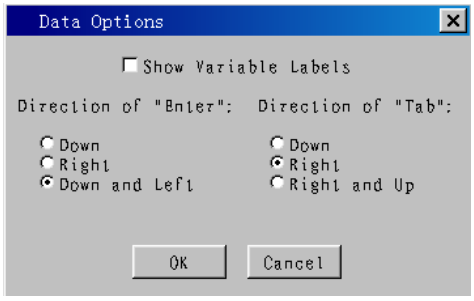


图 1.9: SAS/INSIGHT: 数据窗口选项

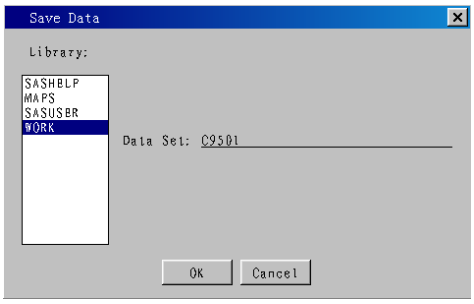


图 1.10: SAS/INSIGHT: 保存数据集

为了保存输入的数据集, 选“File - Save - Data” 菜单, 出现图1.10 那样的保存数据集的窗口, 可以选择数据集放在哪一个数据库, 可以输入一个数据集名, 把这里的A改成C9501, 按OK钮就可以保存数据集。对于比较小的数据集(几个、十几个变量, 几十个观测), 用SAS/INSIGHT的数据窗口可以迅速而直观地输入。SAS/INSIGHT的数据窗口并不象MS Excel那样好用, 它的优势在于其数据探索功能。对于更大量的数据, 一般从其它格式转换或直接访问网络数据库系统。

在数据窗口中如果需要修改某一个值, 需要把鼠标点到其单元格修改后按回车键或制表键移动到其他单元格才能实现修改。在单元格之间移动可以用鼠标单击、制表键、回车、上下光标键等方法。要保存所作的修改还需要用“File - Save - Data”菜单。

当数据窗口中变量较多时, 可以用滚动条滚动窗口内容来查看。如果某个变量比较重要, 可以考虑把它放到第一列的位置, 这只要先单击该变量的名字选中它, 然后在图1.7的菜单中选Move to First。要把某列移到最后, 选中它后用Move to Last菜单。

选中一列只要单击其变量名。如果要选中多个列, 在选中一个后按住Ctrl键单击其它的名字可以添加选中其它变量。选中一个变量后按住Shift 单击另一个变量名可以选中这两个变量及它们之间的所有变量。选中的多个列也可以用Move to First和Move to Last移动。

要选中一个观测(行), 只要单击其观测号(行号)。选多个观测可以用Ctrl 单击或Shift 单击的方法。选

中的观测也可以用Move to First和Move to Last移动到最前或最后。

还可以选中某些列同时选中某些行。只要在后续的选中操作时用添加选中(Shift单击或Ctrl单击)即可。用鼠标在数据窗口数值显示部分拖出一个方框也可以选定一部分数值。

选定了列或者行以后, 用"Edit - Delete"菜单可以删除选定的列或行。用Extract 菜单可以把选定的部分行、部分列或者部分行列取出到另一个窗口。比如, 在SASUSER.CLASS 中先选定所有女生的观测, 再用添加选择（Shift或Ctrl单击）的办法选定NAME和HEIGHT变量, 然后用此命令, 可以打开一个SASUSER.CLASS1 数据窗口, 此数据窗口中只有姓名和身高两列和女生的观测行。可以用主菜单中的"File - Save - Data" 把此新数据集保存为WORK.CLASS1(尽量不用永久数据集存放练习用的数据集)。这样可以由已有数据集挑选部分列、部分行组成新数据集。

要取消所有选定, 只要单击某一单元格而不是行、列标题即可。

下面简单介绍一下数据窗口菜单(图1.7)中各命令:

Find Next — 在选定了若干个观测的情况下, 把下一个被选定的观测显示在窗口第一行的位置。

Move to First — 把选定的行或列移到最前。

Move to Last — 把选定的行或列移到最后。

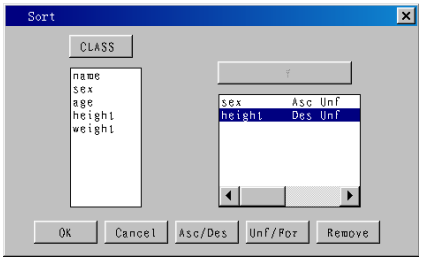


图 1.11: SAS/INSIGHT: 排序对话框

Sort — 在选定了一列的情况下, 把数据集按此列从小到大排序; 在选定了多列的情况下, 按选定的变量次序按这些变量综合排序, 比如在SASUSER.CLASS窗口中先选定SEX然后用Ctrl单击附加选定HEIGHT, 然后排序, 这样的结果是把数据集先按女、男生排序然后在女生内部和男生内部分别按身高从小到大排序。如果没有选定任何列, 则弹出一个对话框(见图1.11), 询问按哪些变量排序, 比如说对SASUSER.CLASS数据, 可以先点SEX, 然后单击Y按钮, 把性别加入了排序变量中, 再选HEIGHT, 单击Y按钮, 把身高作为第二排序变量, 再单击排序变量中的HEIGHT, 单击Asc/Des按钮（这是要求对身高值要由高到低排序）, 按OK后数据将按性别分组, 然后女生、男生内部分别按身高由高到低排列。对话框中的Unf/For按钮选择是否用输出格式表示变量值后排序(Unf不表示, For表示)。

New Observations — 用于快速添加若干个空数据行, 弹出一个对话框要求输入添加的观测数, 缺省是100个。添加的空行中字符型数据先填空值, 数值型数据先填缺失值(用单独的小数点代表)。

New Variables — 用于快速添加若干个新变量。

Define Variables — 设定变量的名字、标签、量测水平、缺省分析用途等, 见图1.8。可以在生成新数据集时定义变量, 也可以对已有数据集的变量属性进

行修改。

Fill Values — 用于自动生成一个等差数列变量。先选定一个数值型变量, 然后用此命令, 将弹出一个对话框要求输入起始值(Value)和增量(Increment), 比如起始值填100, 增量填50, 则此变量的在各观测中的值分别填入为100, 150, 200, . . . 。也可以只选定此变量的一部分行(单击此变量的一个单元格, Shift单击另一单元格就可以选定中间的观测)进行填充。

Extract — 把选定的部分取出到另一个窗口。

Data Options — 本数据窗口的一些设置。弹出对话框如图1.9, 可以选择在数据窗口内按回车时光标是移到下面、左面还是下面最左, 按TAB时光标是移到下面、右面还是右上。选中“Show Variable Labels”可以用变量的标签作为窗口的列标题, 这样有利于理解变量的意义, 但无法知道变量的真实名字。前面说过, 变量标签允许长达40个字符, 允许用汉字。标签可以在数据窗口菜单的Define Variables窗口(图1.8)输入或修改。如果要用较长的汉字标签, 需要适当地用空格分开标签中的汉字以利于分行显示标签。

可以看出, INSIGHT的数据窗口提供了很强的处理数据的功能, 比如输入、修改、查询、排序, 等等。SAS用于数据集管理的一般性界面是VIEWTABLE窗口, 在SAS管理器中双击某一数据集就可以在VIEWTABLE 窗口打开它。VIEWTABLE 用起来没有INSIGHT的数据窗口这么方便, 但是VIEWTABLE能支持巨大的数据集而INSIGHT不行, INSIGHT要把所有数据都读入到计算机内存中才能继续工作, 所以处理不了过大的数据

集。

1.3.3 数据探索— 一维方法

SAS/INSIGHT提供了十分方便的数据探索功能。对一维数据, 可以作直方图、盒形图、马赛克图, 对二维数据, 可以作散点图、曲线图、散点图矩阵, 对三维数据可以作旋转图（三维散点图）。在图上可以选定一些观测, 这些选择结果会同时反映在数据窗口和其它图中。

直方图

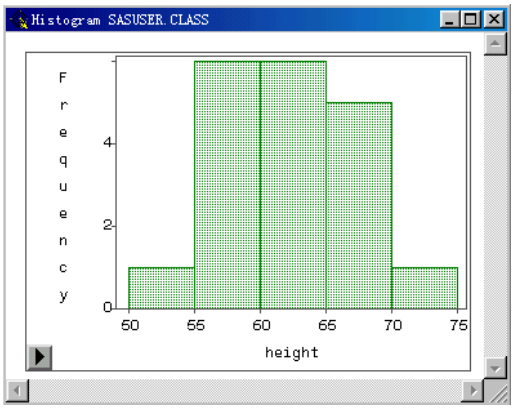


图 1.12: SAS / INSIGHT: 身高的直方图

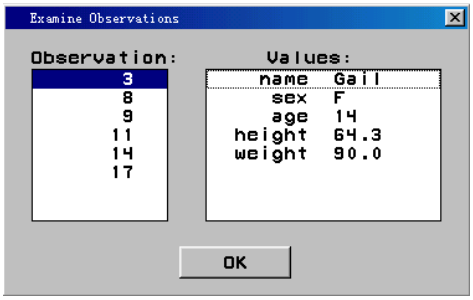


图 1.13: SAS / INSIGHT: 检查观测对话框

以SASUSER.CLASS数据集为例。选定变量HEIGHT, 用“Analyze - Histogram / Bar Charts(Y)”菜单可以打开一个图形窗口生成身高的分布**直方图**, 如图1.12。直方图的每一个条形代表了绘图变量（HEIGHT）在一个区间的取值情况, 比如70到75之

间的条形代表身高在70到75英寸的人, 条形高度为组频数, 即取值在这一区间的观测个数, 可以看出这一组有一个学生。单击这一条形选中在此范围的观测, 可以发现这时数据窗口的相应观测也被选定了, 被选中的是Philip, 身高72英寸。如果双击某一条形, 比如60到65的条形, 就可以在选定相应观测的同时弹出一个检查观测窗口(如图1.13), 窗口中显示各被选中的观测序号, 以及其中一个观测的各变量值。这样可以很方便地检查图中各部分所对应的观测。为取消选定, 只要在图中空白处单击即可。

作出的图形有一个方框包围。如果想改变图形大小, 可以单击方框使其变粗, 然后拖动四个角中的一个, 就可以把图形放大或缩小。甚至还可以把一个角向其对角方向拖动一直拖过对角, 这样可以改变图形的横纵轴方向。拖动边框可以把图形移动到窗口内其它位置。

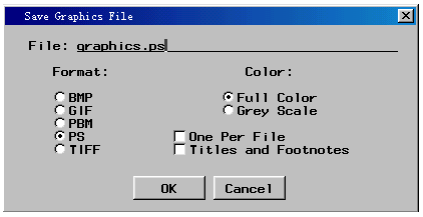


图 1.14: SAS/INSIGHT: 保存图形文件

SAS / INSIGHT 作的图可以保存为外部图形文件。单击图形窗口中图形的边框以选定一个图形, 然后用“File - Save - Graphics File”命令保存为一个图形文件, 见图1.14。现在支持的文件格式有BMP, GIF, PBM, PS, TIFF, 除了要选一种格式外用户在起文件名时还要自己加入适当的扩展名, 比如选了PS格式文件名也要以“.PS”为扩展名。这个对话框中没有显示文件保存在什么目录, 这意味着文件被保存在当前工作目录, 即状态栏右边所显示的目录(见图1.1), 双

击状态栏此处可以更改当前工作目录。

图形中提供了一个设置菜单, 可以单击图形边框角上的向右箭头或在图形内右键单击来打开。菜单内容包括Ticks, 可以设置坐标轴的具体画法；Axes用来指定画不画坐标轴；Observations用来指定是否画出所有的观测, 不选时只对选定的观测画图；Values指定是否标出各条形高度值。

对连续数据（Int型）作直方图可以反映其分布情况, 对离散数据（Nom型）作直方图同样可以反映其分布, 即取每一个离散值的比例大小（频数分布）。比如, 在作了身高的直方图后, 选定变量SEX, 对其作直方图, 则结果打开一个新图形窗口作出只有两个条形的条形图, 一个标记为F, 另一个标记为M, 高度分别为9和10, 即有9个女生, 10个男生, 男女比例为10：9。单击标F的条形, 可以看到数据窗口中所有女生的观测被选定, 另外还可以看到已作的身高的直方图也发生了变换, 身高的每一个条形都分成了颜色不同的两部分, 其中下面的一部分代表女生。在一个窗口中选定部分观测就可以在所有窗口中选定相应观测是SAS / INSIGHT 强大的数据探索功能的一个突出特色。

在用Analyse菜单中的作图命令作图时如果没有选定的变量则弹出一个对话框提问用哪一个变量作图, 如果对身高作图, 只要选HEIGHT然后按Y钮即可。

盒形图

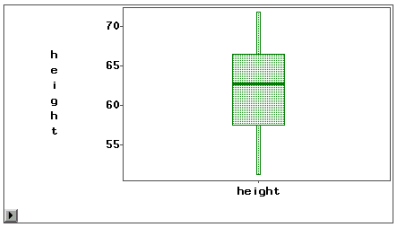


图 1.15: SAS/INSIGHT: 身高的盒形图

盒形图是另一种表现数值型变量分布的图形。比如, 要画身高分布的盒形图, 选定变量HEIGHT然后用“Analyze - Box Plot / Mosaic Plot”可以作出图1.15。从图形菜单中选Values可以标出图中重要数据值。可以看出, 此盒形图的横轴没有用处, 纵轴代表身高的取值范围。盒形的中间有一条粗线, 这是身高分布的中位数的位置, 盒子上边线是分布的四分之三分位数, 下边线是分布的四分之一分位数, 盒子上下边线包含了分布的中间50%的观测。盒子的长度叫做分布的四分位间距, 其作用类似于标准差, 可以反映数据分布的分散程度。从盒子边线向外画了两条线叫做触须线, 最长可以延伸到四分位间距的1.5倍, 但是如果已经到了数据的最小值或最大值处就不再延伸。如果触须线没有达到数据的极端值, 则这些数据点用触须线以外的点来画出, 一般认为这样的点是异常点。从盒形图可以看出数据的偏斜情况, 比如我们看到盒子的下半部比上半部长, 而且下触须线比上触须线长, 说明身高分布略左偏。

用盒形图菜单中的“Means”选项可以在盒形图上加画一个菱形, 菱形的中间代表分布的平均值, 菱形端点到中间距离为两倍标准差。如果是变量服从正态分布, 菱形上下端点之间应该包含大约95%的观测。平均值和中位数的比较也能反映变量的偏斜情况, 平均值低于中位数可能左偏。

单击或双击盒形图的某一部分(盒子上半部或下半部、触须线、极端值)可以选定部分观测。

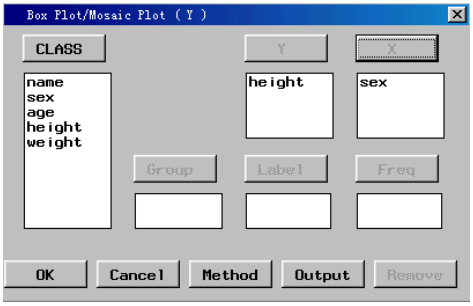


图 1.16: SAS/INSIGHT: 盒形图变量指定

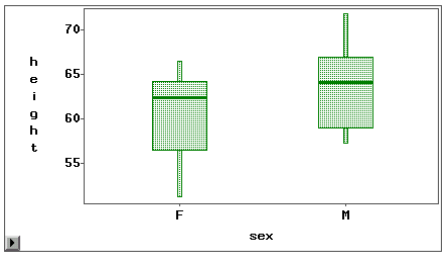


图 1.17: SAS/INSIGHT: 男女身高的盒形图比较

盒形图可以方便地比较按某分组变量分组后的分布情况。比如, 如果我们想看一看男女的身高分布有何异同, 不选任何变量启动“Analyze - Box Plot / Mosaic Plot”菜单, 弹出选择变量的对话框如图1.16, 选身高为Y变量, 选性别为X变量, 画出的图见图1.17。图中有两个盒形图, 女生一个, 男生一个。从图中看出, 男生身高普遍高于女生, 且女生身高分布左偏较男生严重。这种并排盒形图可以十分直观地比较两个相关的分布。作盒形图时指定多个Y变量也可以作出并排的盒形图, 比如, 同时指定身高和体重作为Y变量作盒形图就可以生成身高和体重的并排的盒形图。

马赛克图

Analyze菜单的“Box Plot / Mosaic Plot”命令对连续型变量作盒形图, 对离散型变量将作**马赛克图**。比如, 对性别变量作图得图1.18。选“Values”菜单后

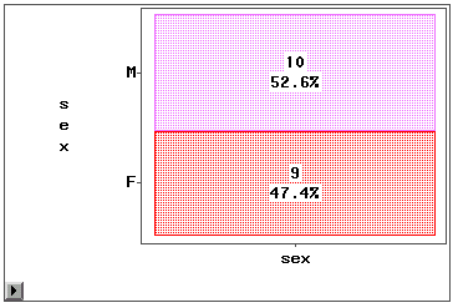


图 1.18: SAS/INSIGHT: 性别的盒形图

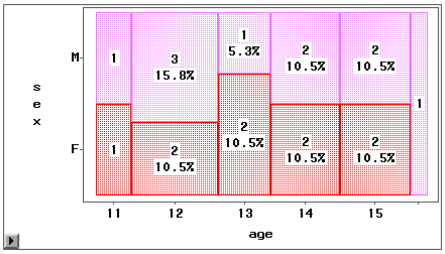


图 1.19: SAS/INSIGHT: 性别和年龄的交叉马赛克图

标出了男女的人数、百分比。马赛克图一般不对单个变量作, 而是对两个离散变量来作。比如, 先把SASUSER.CLASS中变量AGE的量测水平由Int改为Nom, 然后取消所有变量的选定, 启动“Box Plot / Mosaic Plot”, 选SEX为Y变量, 选AGE为X变量, 作图如图1.19。这种图的好处是直观显示了两个变量每种取值组合的观测个数和比例。单击其中一个方块可以迅速选中一个分组, 比如单击击年龄为11性别为女（F）的方块可以看到这一组的学生。双击某一方块可以检查相应组的学生。

1.3.4 数据探索— 二维

SAS/INSIGHT可以作曲线图、散点图、散点图矩阵, 可以在散点图中刷亮观测。

曲线图

曲线图有一个取值由小到大的X变量, 有一个或几个Y变量, 以X变量为横坐标对Y变量画曲线。为

了演示曲线图, 打开SASUSER.AIR数据集(用“File - Open”菜单)。这个数据集是德国某城市一周时间每小时记录的空气污染情况。变量DATETIME是记录的日期时间, 为特殊SAS格式数据, 变量DAY为星期几, HOUR为几点钟, CO、O3、SO2、NO、DUST分别为一氧化碳、臭氧、二氧化硫、一氧化氮、粉尘的浓度, WIND为风速。要画一氧化碳的曲线图, 可以在未选任何变量的情况下用“Analyze - Line Plot”, 弹出变量对话框(图1.20), 选DATETIME为X变量, CO为Y变量, 可以画出CO的时间序列曲线图, 见图1.21。单击曲线上某一个点可以显示其观测序号, 双击可以检查观测。如果想单击曲线上点时不显示观测序号而显示记录时间是几点, 可以在曲线图窗口中选主菜单的“Edit - Windows - Renew”, 可以再次弹出变量窗口, 选HOUR并按Label钮把时间指定为标签变量。如果一开始就已经把HOUR作为Label使用(在数据窗口HOUR上方左边小格单击可以出现Group, Freq, Label, Weight等特殊作用的选择)。这时在作的CO的曲线图上单击一个点显示的就是记录时间了。可以看出CO的高峰一般在早晨8点和晚上17点–21点。选中图形菜单(右键或单击向右三角)中的Observations可以画出所有数据点的符号, 否则只有选中的观测画点(不影响曲线)。

可以在图上同时画出多条曲线。比如, 想考察风速对污染的影响, 在图形窗口中再用主菜单的“Edit - Windows - Renew”, 把WIND也作为Y变量, 画出的图就有两条不同颜色的曲线, Y轴的变量标签也使用了与曲线相对应的颜色。单击变量符

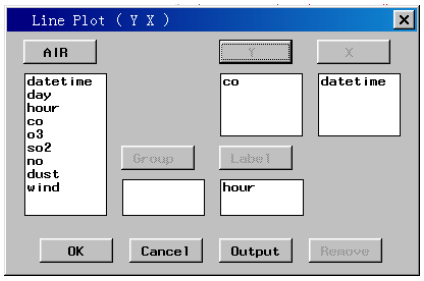


图 1.20: SAS / INSIGHT: 曲线图变量选择框

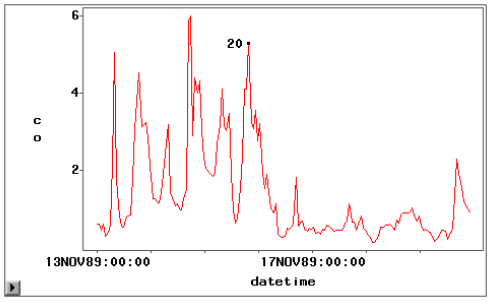


图 1.21: SAS / INSIGHT: 一氧化碳浓度曲线图

号CO或WIND可以加重显示对应的曲线以区分这两条曲线。见图1.22。图中被选的点是风速的最高值, 时间是11点。注意观测在一条曲线中被选在另一条曲线中也被选。从此图可以看出风速对污染有较明显的影响, 风大时污染较轻。

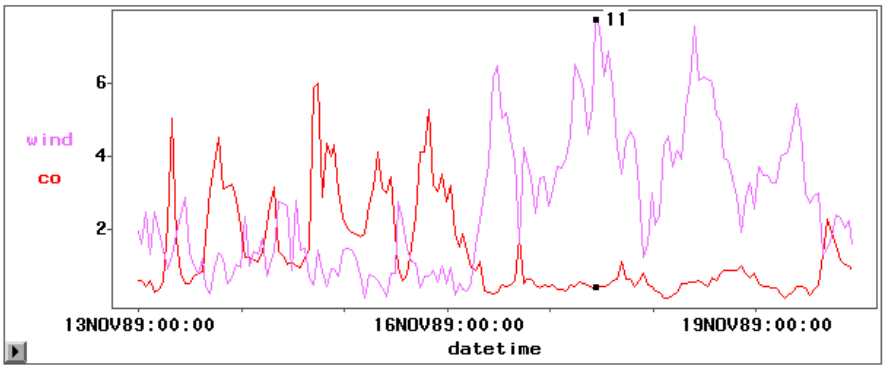


图 1.22: 一氧化碳浓度和风速

散点图

散点图也有一个X变量和一个Y变量, 但不要求X变量有从小到大的次序, 画图不用连线而是用散点画出

每一对X、Y坐标。比如对SASUSER.CLASS, 我们希望通过画图了解身高和体重的关系。在数据窗口中先选定体重(Y轴变量)再附加选定身高(X轴变量), 启动菜单“Analyze - Scatter Plot”, 就可以生成以体重为纵轴以身高为横轴的散点图(见图1.23)。从图可以看出体重与身高有明显的线性相关关系。

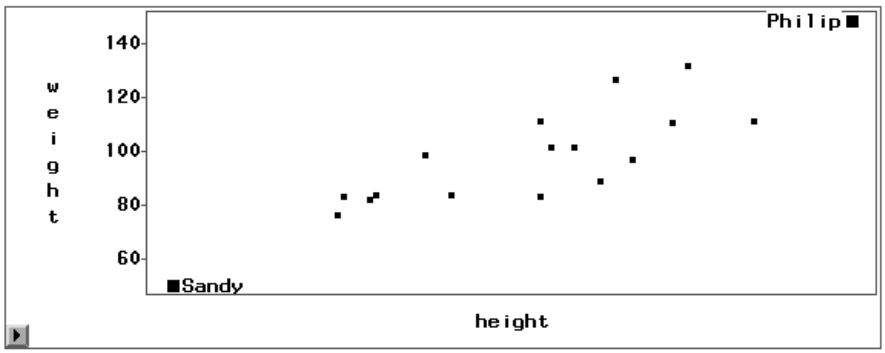


图 1.23: 体重对身高的散点图

为了解哪一个点代表哪一个学生, 单击一个点可以显示其观测序号, 双击可以检查观测。为了在单击时可以显示学生名字而不是观测序号, 需要把NAME指定为标签变量。这可以在生成散点图时先不在数据窗口选X、Y变量而是直接启动“Analyze - Scatter Plot”菜单, 弹出变量对话框, 在其中选X、Y变量并把NAME指定为Label变量。或者预先在数据窗中把NAME指定为Label变量也可以。这时, 单击散点图中最左下角的那个点可以显示名字Sandy, 单击最右上角的那个点可以显示Philip。选多个点可以用附加选中的办法(Shift或Ctrl单击)。

为了在散点图中选定多个点, SAS/INSIGHT还提供了一种称为“**刷亮**(Brushing)” 的操作。在图中拖动

鼠标光标可以拖出一个小长方形, 在这个长方形中的点都被选中, 称它为刷子。选中的点在数据窗口也被选中, 可以在数据窗口翻页查看, 或用数据窗口的Find Next菜单命令查看, 或在数据窗口用Move to First菜单命令把选中的点移到最前查看。双击长方形(刷子)可以弹出检查观测窗口, 在那里可以逐个查看选中的观测内容。

拖动刷子的角可以改变其大小。拖动刷子内部可以移动它的刷亮位置, 使进入刷子的点被选中, 而离开了刷子的点被取消选中。可以同时用附加选中(Ctrl单击)的办法加选不在刷子内的点, 这些点还可以显示标签。在拖动刷子时如果同时按住Shift或Ctrl键则为附加选定, 即进入刷子的点被选中而离开刷子的点仍保持被选中, 否则拖动刷子时只有刷子内的点被选中。可以按住Shift或Ctrl键拖出第二个刷子, 这时第一个刷子不再显示但它刷亮的点仍保持刷亮, 移动第二个刷子时如果按住Shift或Ctrl键仍可保持已有选定。为了取消所有选定, 只要点击图内空白处。

散点图矩阵

散点图矩阵画出多个变量两两间的散点图以考察多变量关系。以SASUSER.CLASS 为例, 比如说我们想了解年龄、身高、体重间的关系。先把年龄的量测水平设为连续型(Int), 在数据窗口选定年龄、身高、体重, 可以作出图1.24。我们看到三个变量两两组合有三种组合, 每种组合有两个图形(横纵轴对换)。散点图矩阵对角线为变量标记和变量取值范围, 该变量

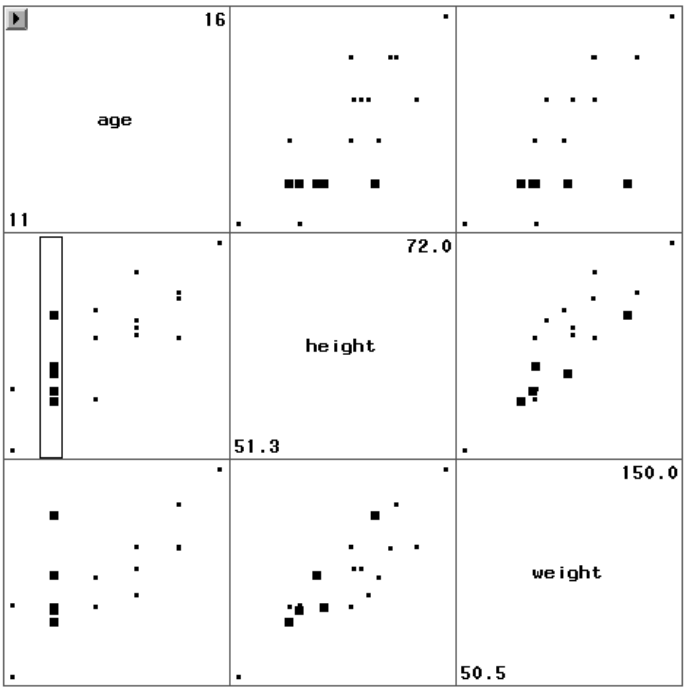


图 1.24: 散点图矩阵

是其所在行的纵轴变量, 是其所在列的横轴变量。比如第二行第一列的图纵轴变量是HEIGHT, 横轴变量是AGE, 为身高对年龄的散点图。其对称位置(第一行第二列)是年龄对身高的散点图, 两者只是把横纵坐标旋转对调。

散点图矩阵不仅可以同时看到多个散点图, 在一个散点图中被选中的点在其它散点图和数据窗口中也同时被选中。这样, 我们可以在一个图中选一个极端点, 看它在其它图中是否也处于极端位置。在一个散点图中刷亮的点在其它散点图中也同时被刷亮, 这样, 我们可以观察, 年龄和身高都比较小时, 体重是否也比较低。可以移动刷子, 同时其它散点图中被选中的点也在变化。从图1.24 可以看出, 年龄由小到大变

化时身高、体重一般也变大, 但同一年龄的学生的身高、体重差距较大。

SAS/INSIGHT提供了自动移动刷子的功能。在拖动刷子时松开鼠标按钮, 类似于“抛出”刷子, 刷子就可以按抛出的方向继续移动并反弹。不过现在还较难控制自动移动的速度, 有时移动过快。

1.3.5 数据探索— 三维

SAS/INSIGHT对三维数据可以作称为旋转图的三维散点图。我们用如下的程序生成了一个模拟的数据集SASUSER.CLUS:

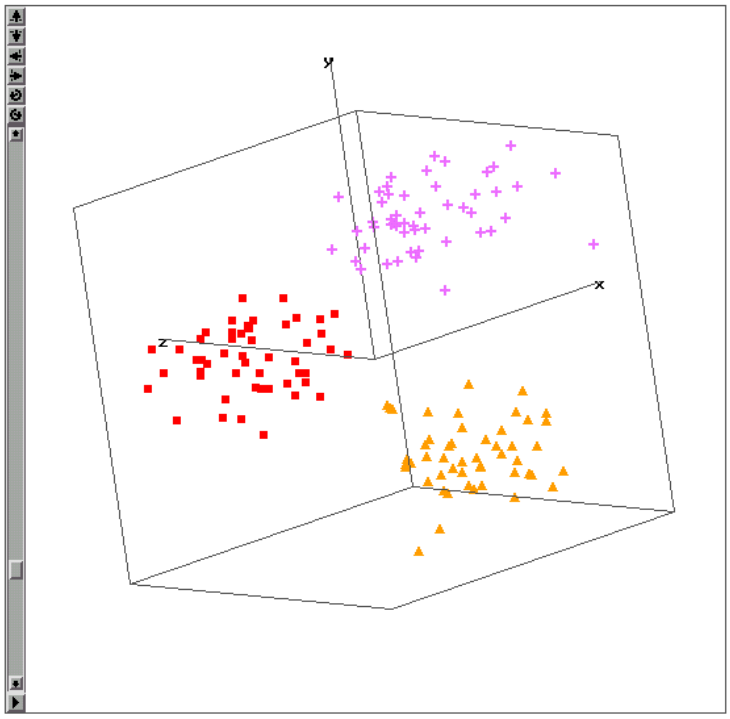


图 1.25: SAS/INSIGHT: 旋转图

```
data sasuser.clus;
  do i=1 to 50;
```

```
    g='a';
    x = normal(0); y = normal(0);
    z = normal(0);
    output;
  end;
  do i=1 to 50;
    g='b';
    x = 3 + normal(0); y = 4 + normal(0);
    z = -3 + normal(0);
    output;
  end;
  do i=1 to 50;
    g='c';
    x = 3 + normal(0); y = -4 + normal(0);
    z = -3 + normal(0);
    output;
  end;
  drop i;
run;
```

生成的数据集中包括X, Y, Z三个数值型变量, 在数据窗口依次选定Z, Y, X, 然后启动菜单“Analyze - Rotating Plot”, 可以生成一个三维散点图。图1.25是经过旋转并根据分组添加颜色和符号的图形。

这种三维散点图之所以称为旋转图, 是因为坐标系可以在三维空间绕原点任意旋转。图形的左侧有一个小工具栏, 其中有向上、下、左、右、逆时针、顺时针旋转的图标, 再往下有一个滚动条, 用它来规定自动旋转的速度(越靠上边旋转越快)。左下角是图形的菜单(向右的三角形)。

为了旋转坐标系, 单击左侧的旋转方向图标。按住旋转图标可以连续旋转。按住Shift 或Ctrl 再旋转可以实现自动旋转。当鼠标光标移到图形的四个角时光标形状变成了手的形状, 单击可以旋转, 拖动可以

连续旋转, 拖动时“抛出”可以自动旋转。自动旋转中可以随时拖动图形以改变旋转方向。

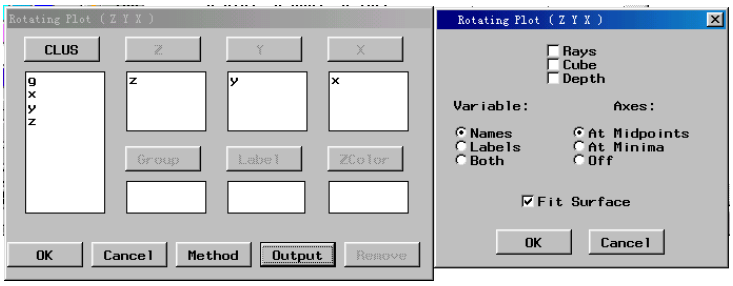


图 1.26: SAS/INSIGHT: 旋转图定制

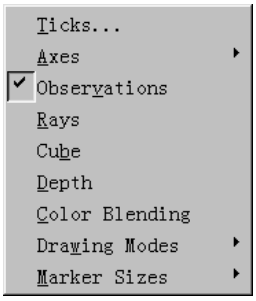


图 1.27: SAS / INSIGHT: 旋转图菜单

旋转图还可以把各散点用一个曲面来拟合, 相当于拟合一个二元函数$z = f(x, y)$。这只要在开始要求画旋转图的界面(见图1.26) 中按Output钮选中其中的Surface。如果已经画了图, 可以调用主菜单的“Edit - Windows - Renew”来显示这个对话框。

旋转图的菜单(图1.27)中, Ticks用来调整坐标轴刻度, Axes可以选坐标轴以数据中心点为原点、以最小值为原点、画在适当位置, 还是不画坐标轴。Observations指定画出所有观测, 如果没有选中此项则只画被选中的观测。Rays从原点向每个散点画射线。Cube在散点四周画一个长方体盒子。Depth可以使离视点近的点画得较大, 离得远的点画得较小。在画拟合曲面图时, Color Blending 可以把除x、y、z 变量外的一个变量作为颜色画在坐标系中, 用不同颜色代表该变量的不同值。Drawing Mode 选择与画图或上颜色有关的选项。Markers Sizes选择散点的大小。

图 1.28: SAS / INSIGHT: 工具窗口

对三维数据SAS / INSIGHT 还可以作等值线图(Contour), 这里不再举例说明了。

1.3.6 图形的调整

SAS/INSIGHT提供了很强的调整绘制的图形的功能。比如, 调整坐标轴的画法, 点的大小、符号、颜色, 隐藏某些观测, 等等。

给不同观测使用不同的符号和颜色画点有助于迅速区分不同类观测的特点。比如, SASUSER.IRIS 数据集中包含了Fisher著名的Iris数据, 其中有三种不同的鸢尾属植物的花瓣、花萼长、宽的测量数据, 希望从这些测量数据找出区分这三种植物的指标。

为了直观看到不同植物的测量数据的特征, 最好用不同颜色画每一种植物的散点。打开数据集后, 选定分类变量SPECIES, 调用“Analyse - Box Plot - Mosaic Plot”菜单来作其马赛克图, 可以看到此变量的三个值为Virginica、Versicolor、Setosa。用“Edit - Windows - Tools” 菜单可以打开一个工具窗口, 如图1.28。这个窗口可以改变观测符号的颜色、符号, 连线的线型、线宽, 可以放大图形局部。在打开的马赛克图中先选定Virginica, 这时所有类型Virginica的观测被选中, 按一下工具窗口中的红色, 就给所有这些观测规定了绘图符号为红色。类似指定Virsicolor为绿色, Setosa为蓝色。作PETALWID (花瓣宽) 对PETALLEN (花瓣长)的散点图, 可以作出三种不同植物用不同颜色绘点的散点图, 见图1.29。

利用一个变量的不同值来确定观测绘点的颜色还可以自动进行, 方法是先选定该变量（如SPECIES）, 然后单击工具窗口的渐变颜色棒, 就可以为SPECIES的每一不同值分配一种不同颜色。这一方法不仅适用于SPECIES这样的名义变量, 也适用于数值型变量。颜色棒的颜色可以调整, 比如要把颜色棒变为由红到蓝, 只要把红色方块拖到颜色棒左端, 把蓝色方块拖到颜色棒右端。

为了改变绘点符号的大小, 调用图形菜单（图形边角上的向右三角符号）中的Marker Sizes菜单可以选择一个合适的符号大小。

除了用不同颜色来区分不同种类的观测外, 还可以用不同的符号来画不同的观测。比如, 选定SPECIES为Virginica的观测后, 单击工具窗口的菱

形图标把此类观测的绘点符号变为菱形。类似指定Virsicolor用三角, Setosa用倒三角号, 作的散点图见图1.29。从图中可以看出, 用倒三角绘制的Setosa类和其它两类差别很大, 单靠花瓣的长、宽就可以把这一类与其它两类区分开, 但是用菱形绘制的Virginica类和用三角绘制的Virsicolor类则在能大体区分开的同时有少数观测混杂在一起, 所以单靠花瓣的长、宽测量数据不能把这两类很好地区分开。图1.25也是一个用了不同颜色和符号的图, 可以看出X, Y, Z 三个变量可以区分开变量G所定义的三个类。

利用一个分类变量来决定不同的绘点符号除了上述的对每一类观测分别选定, 然后指定绘点符号的办法, 还可以选定这一分类变量, 然后单击工具栏中绘点符号下面的多种符号的长棒形图标, 可以自动为每一类分配一个绘点符号。

不同类观测用不同的颜色和符号来绘点是一种强有力的数据探索手段, 恰当使用可以直观地发现不同类型观测的区别。

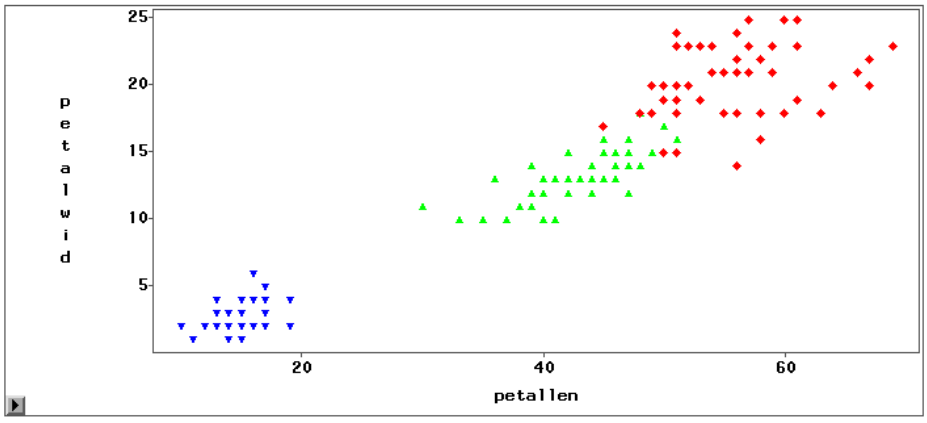


图 1.29: 不同颜色和符号的观测

1.3.7 分布研究

我们得到数据之后, 通常需要先检查一下数据中每个变量的分布情况才开始进行更深入的分析。对于变量分布, 我们关心的是变量的取值类型(是连续取值还是离散取值, 是分类的、有序的还是区间的)和变量的分布规律。所谓*分类变量*是指变量离散取值而且代表不同的分类, 如性别、职业。*有序变量*也是离散取值的, 但是变量值之间有大小次序, 比如年龄组包括青少年、中年、老年。*区间变量*指本质上可以取到某个区间内任何一个值的变量, 如身高。对于离散变量, 分布规律是变量值落在各个可能值上的比例规律。对于区间变量, 分布规律最好是用其分布密度描述, 但是分布密度是一条理论上有无穷多点的曲线, 所以也需要利用一些更简单的描述方式, 比如说明变量取值的范围, 在什么值附近取值可能性大, 什么值附近取值可能性小, 变量取值以什么为中心, 变量取值集中和分散的程度如何, 变量分布是否偏向某一方, 变量是否异常值多, 等等。

SAS/INSIGHT提供了很强的一维分布研究功能。对连续型变量, 除了可以画直方图、盒形图外, 还可以作各种统计表, 比如矩、分位数表, 可以在直方图上画拟合密度曲线, 可以检验分布是否来自正态、对数正态、指数、威布尔分布, 等等。对离散型变量, 可以画马赛克图、条形图、频数表。

为了研究SASUSER.CLASS中身高的分布, 在未选中变量的情况下, 启动"Analyze - Distribution(Y)"菜单, 出现一个选择变量对话框, 选Y变量为HEIGHT(先选中HEIGHT再按Y按钮), 按OK可以

打开一个新窗口, 显示身高的直方图、盒形图、矩统计量表(图1.30)、分位数表（图1.31）。这里的直方图和直接用“Histogram” 菜单生成的直方图略有区别, 其纵轴代表的是密度值而不是频数值。

Moments

N	19.0000	Sum Wgts	19.0000
Mean	62.3368	Sum	1184.4000
Std Dev	5.1271	Variance	26.2869
Skewness	-0.2597	Kurtosis	-0.1390
USS	74304.9200	CSS	473.1642
CV	8.2248	Std Mean	1.1762

图 1.30: SAS / INSIGHT: 矩统计量表

Quantiles

100% Max	72.0000	99.0%	72.0000
75% Q3	66.5000	97.5%	72.0000
50% Med	62.8000	95.0%	72.0000
25% Q1	57.5000	90.0%	69.0000
0% Min	51.3000	10.0%	56.3000
Range	20.7000	5.0%	51.3000
Q3-Q1	9.0000	2.5%	51.3000
Mode	62.5000	1.0%	51.3000

图 1.31: SAS / INSIGHT: 分位数表

各统计量是SAS中经常使用的, 我们在此加以说明。设变量为Y, 各观测值为$y_1, y_2, \cdots, y_n$。有时每个观测还带一个加权w_i, $i = 1, 2, \cdots, n$, 在没有指定加权变量时认为加权恒为1。

N — 观测个数n

Sum Wgts — 加权和$\sum_{i=1}^{n} w_i y_i$

Mean — 均值$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$

Sum — 总和$\sum_{i=1}^{n} y_i$

Std Dev — 标准差$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (y_i - \bar{y})^2}$

Variance — 方差$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (y_i - \bar{y})^2$

Skewness — 偏度$\frac{n}{(n-1)(n-2)} \Sigma \left(\frac{y_i - \bar{y}}{s} \right)^3$

Kurtosis — 峰度$\frac{n(n+1)}{(n-1)(n-2)(n-3)} \frac{\sum (y_i - \bar{y})^4}{s^4} - \frac{3(n-1)^2}{(n-2)(n-3)}$

USS — 加权平方和$\sum_{i=1}^{n} w_i y_i^2$

CSS — 加权离差平方和$\sum_{i=1}^{n} w_i (y_i - \bar{y})^2$

CV — 变异系数$\frac{s}{\bar{y}} \cdot 100$

Std Mean — 均值的标准误差$s/\sqrt{n}$

其中加权的常见情形是当一个观测实际代表完全相同若干个样品时, 求和、平方和等都要加权。比如, 第i个观测代表w_i个样品时, 求变量Y的真正总和就需要用加权公式$\sum_{i=1}^{n} w_i y_i$。偏度可以表现变量分布的偏斜, 负值为左偏, 正值为右偏。峰度表现变量分布与正态分布相比是重尾(分布密度在正负无穷处衰减缓慢) 还是轻尾(分布密度在正负无穷处衰减迅速)。

标准误差在统计中是一个十分重要的概念, 它代表把估计量看成随机变量时其标准差的估计, 这里的Std Mean是均值的标准差的估计, 实际计算公式是$s/\sqrt{n}$, 而均值的理论标准差为$\sigma/\sqrt{n}$。如果估计量服从正态分布, 通常用估计量加减两倍标准误差作为估计量的置信区间。

分位数表中, Max是最大值, Q3是四分之三分位数, Med是中位数（反映数据中心位置）, Q1是四分之一分位数, Min是最小值, Range是最大值减最小值, Q3 − Q1为四分位间距, 可以反映数据取值分散程度, Mode是众数, 即出现最多的值。

在打开了身高分布的窗口之后主菜单中的Tables、Graphs、Curves菜单被开放。在Tables菜单中可以选加一些统计表, 比如Basic Confidence Intervals可以计算均值、标准差、方差的各种置信度的置信区间, Tests for Location用于检验均值为某常数值(一般是0)的假设, 可以用t检验、符号检验、符号秩检验, Frequency Counts是频数表, 为每一观测值

的频数、百分比、累计频数。

Robust Measures of Scale给出了变量分布分散程度的五种稳健估计, 包括四分位间距, Gini's Mean Difference, MAD, Mn, Qn, 在给出统计量值的同时还给出了用这些统计量得到的标准差稳健估计。Gini's Mean Difference 计算公式为$g = \frac{1}{n(n-1)/2}\sum_{i<j}|y_i - y_j|$, 对于正态分布其期望值为$2\sigma/\sqrt{\pi}$, 可以据此估计标准差。

Tests for Normality 给出四种正态分布检验, 包括Shapiro-Wilk, Kolmogorov-Smirnov, Cramer-von Mises, Anderson-Darling。

"Trimmed/Winsorized Mean" 计算去掉(或替换)最大和最小若干个值后的平均值, 是一种稳健的均值估计。

在Graphs菜单中已选了直方图、盒形图, 用"QQ Plot"菜单可以作QQ图, 即分位数–分位数图。常用的QQ图是相对于正态分布的QQ图, 一般作法如下：设有n个观测$y_1, y_2, \ldots, y_n$, 并已从小到大排列, 则y_i是总体的i/n分位数的估计, 设x_i是标准正态分布的i/n分位数, 则在样本来自正态$N(\mu, \sigma^2)$的情况下用(x_i, y_i) $(i = 1, 2, \ldots, n)$作为坐标画散点图应该近似呈现为截距μ、斜率σ的一条直线。但是, 这样的画图有一个缺点：y_1是观测到的最小值, 对应于$1/n$分

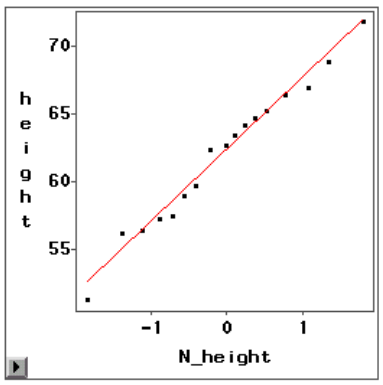


图 1.32: SAS / INSIGHT: 身高的正态QQ图

位数, y_n是观测到的最大值, 却对应于n / n = 100%分位数, 对最小和最大值的处理不对称, 相当于说总体分布不能超过y_n, 这是不合理的。所以在实际画正态QQ图时, y_i不是对应于标准正态的i/n分位数而是对应于其$(i-0.375)/(n+0.25)$分位数, 这种做法叫做连续性修正。这时, y_1对应于$\frac{0.625}{n+0.25}$分位数而y_n对应于$1-\frac{0.625}{n+0.25}$分位数, 两边各留了$\frac{0.625}{n+0.25}$。图1.32为身高的正态QQ图, 其中画出了班上19个学生的19个点, 每个点的纵坐标为变量值, 而横坐标为该值的累计百分比频率经连续性修正后对应的标准正态分位数。QQ图的各种不同形状能够反映出变量分布的偏斜情况和重、轻尾情况。在QQ图中也可以选观测、刷亮等。画出QQ图后选主菜单中的"Curves - QQ Ref Line" 可以为图中散点画一条拟合直线。

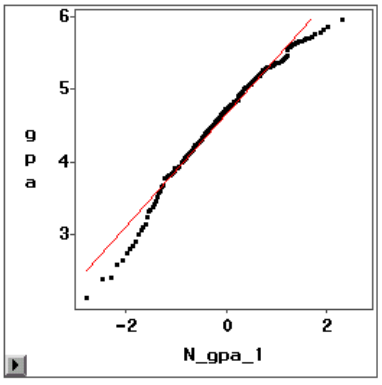


图 1.33: SAS / INSIGHT: GPA的正态QQ图

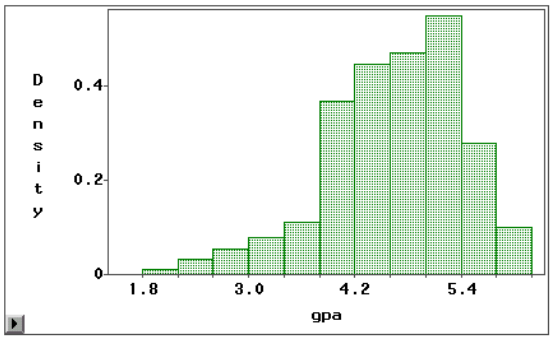


图 1.34: SAS / INSIGHT: GPA的直方图

图1.32的身高的QQ图显示身高基本服从正态分布。如果我们画SASUSER. GPA 中GPA 分数的QQ图(图1.33), 就可以看到GPA的分布呈现左偏的

情况。这是因为, 在QQ图的左下端, GPA散点的走向比正态（图中直线）偏下, 说明GPA分布的左尾比正态长；在QQ图的右上端, GPA散点的走向比正态偏右下, 说明GPA分布的右尾比正态短, 即分布左偏。作为验证, 可以看一看的图1.34直方图。

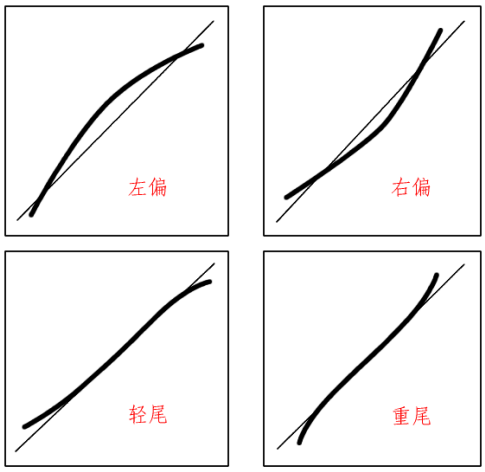


图 1.35: SAS / INSIGHT: 左偏、右偏、轻尾、重尾的典型正态QQ图

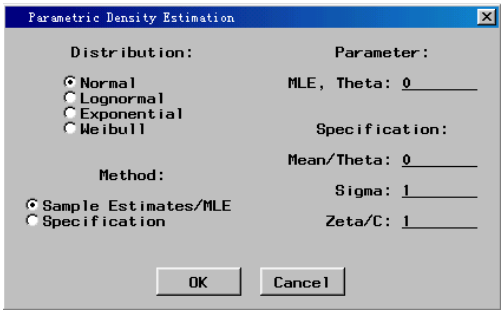


图 1.36: SAS / INSIGHT: 参数密度估计

图1.35给出了与正态相比左偏、右偏、轻尾、重尾的分布的QQ图的典型模式。

除了可以作正态分布QQ图外, 还可以作对数正态、指数分布、威布尔分布的QQ图。对数正态要指定参数Sigma, 威布尔分布要指定形状参数C。

SAS/INSIGHT为研究一维变量分布除画直方图外还提供了两类**分布密度估计**：参数估计和非参数估计。参数估计可以拟合正态、对数正态、指数、威布尔分布密度。非参数估计使用核估计。

比如, 为了估计身高的正态密度并把密度曲线叠加在直方图上, 选“Curves - Parametric Density”, 弹出对

话框图1.36, 指定正态分布且方法为用样本估计分布密度参数。按OK后作出的图见图1.37。

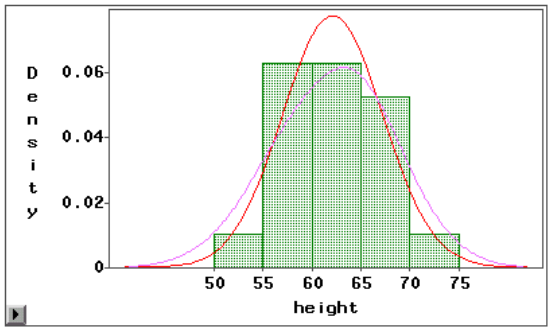


图 1.37: SAS / INSIGHT: 叠加了正态密度和核密度估计的直方图

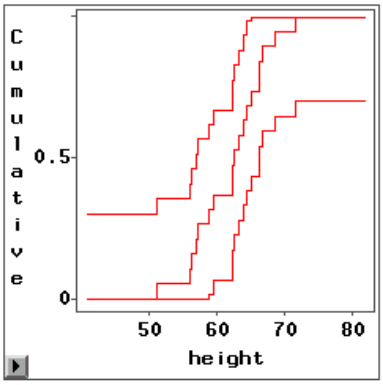


图 1.38: SAS / INSIGHT: 经验分布函数及95%置信限

为了作身高密度的核估计图, 选“Curves - Kernel Density”, 弹出一个对话框, 可以选三种核函数：正态核、三角核、二次函数核, 可以自动拟合最优的密度估计(方法为AMISE)或者自己指定平滑参数C。见图1.37。

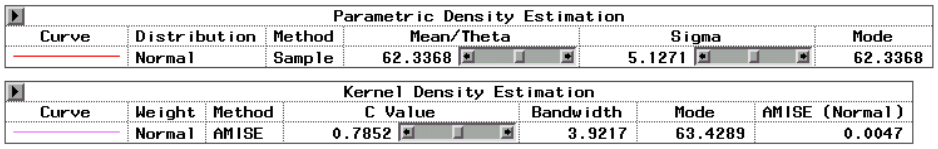


图 1.39: 密度估计参数表

作了密度曲线图后在图形下面将出现显示密度估计主要参数的表格, 见图1.39。单击其中的曲线标志可以加亮显示图中的曲线。对参数密度估计, 给出了估计的参数, 比如正态的均值、方差；对核估计, 给出了核函数类型, 及平滑参数值。有些参数旁边有一

个滑块, 可以手工选择参数的值。比如拖动核估计中的平滑参数, 此参数变小时估计的曲线变粗糙, 变大时曲线变光滑。

在“Curves”菜单中还提供了对样本经验分布函数的估计。选“Curves - Empirical CDF”即绘制样本经验分布函数。选“Curves - CDF Confidence Band”并选一个置信限可以在经验分布函数两边画分布函数的置信限, 见图1.38。

用经验分布函数估计分布函数相当于用直方图估计分布密度。分布函数也可以用参数分布函数（如正态分布）来估计。选“Curves - Parametric CDF”并选分布类型可以画出估计的分布函数。

Tests for Normality

Test Statistic	Value	p-value
Shapiro-Wilk	0.979083	0.9312
Kolmogorov-Smirnov	0.144272	>.1500
Cramer-von Mises	0.040516	>.2500
Anderson-Darling	0.235778	>.2500

图 1.40: SAS / INSIGHT: 身高的正态性检验

Tests for Location: Mu0=0

Num Obs != Mu0:19

Num Obs > Mu0:19

Test	Statistic	p-value
Student's t	53.00	<.0001
Sign	9.50	<.0001
Signed Rank	95.00	<.0001

图 1.41: SAS / INSIGHT: 均值的检验

SAS / INSIGHT 的Curves菜单提供了分布检验。选“Curves - Test for Distribution”, 可以进行正态分布、对数正态分布、指数分布、威布尔分布的检验。

说明：对身高进行正态性检验得到了图1.40的结果。它给出了四种正态性检验的统计量及相应的p值(零假设是总体服从正态分布)。对于图中身高的正态性检验因为p值较大所以检验结果不显著, 不能否定正态性假设。

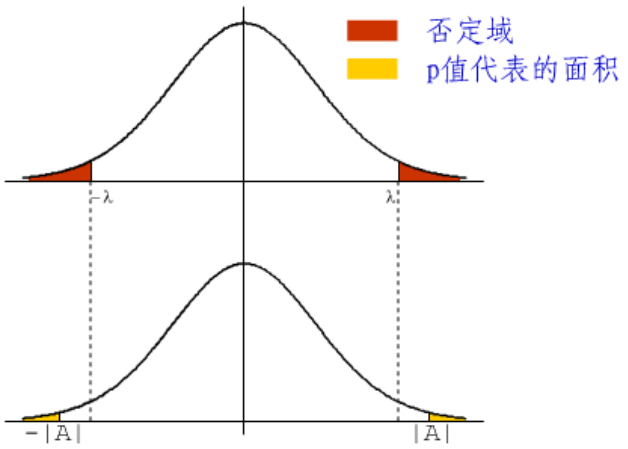


图 1.42: p值的图示。可以看出p值小于检验水平α与检验统计量值A落入否定域$\{|A| > \lambda\}$是等价的。

在SAS中, 统计假设检验的结果一般用检验的**p值**给出。这与我们习惯的做法稍有不同, 以单正态总体的均值检验为例。假设我们要检验SASUSER.-CLASS中学生的身高是否均值为零(这当然不可能, 我们为简单起见用这种假设), 设总体服从$N(\mu, \sigma^2)$, 要检验的零假设为$H_0 : \mu = 0$, 水平0.05, 统计量使用t统计量$t = \frac{\bar{y}-0}{s/\sqrt{n}}$, 一般我们用的假设检验方法定否定域为$W = \{|t| > C\}$, 其中$C$为$n-1$自由度t分布的双侧0.05分位数($\Pr\{|t| > C\} = 0.05$), 当用样本算出的t统计量的值(如t=A)落入否定域时($|A| > C$)否定零假设。在SAS中不需要这样指定否定域, 它可以先用样本计算出t统计量的值(A), 如果这个A绝对值很大就否定零假设。t统计量绝对值值是不是很大可以用这样一个$p = \Pr\{|t| > |A|\}$来衡量, p是一个0到1之间的数值, 显然$|A|$越大, p越小, $p < 0.05$与$|A| > C$是

等价的。所以, 如果p小于0.05, 就否定零假设, 称检验结果是显著的。否则不否定零假设。见图1.42。

对SASUSER.CLASS中HEIGHT变量, 在其分布窗口中选菜单“Tables - Tests for Location”, 输入要检验的均值为0, 得到的结果见图1.41。计算得到的t统计量值为A＝52.9971, p值为Pr{$|t| > 52.9971$}小于0.0001。因p值小于0.05所以结果是否定零假设, 结论是身高均值不为零。

SAS/INSIGHT还提供了曲线拟合、回归、logistic回归、Poisson回归、相关分析、主成分分析等高等统计功能, 我们后面再陆续介绍。

练习

1. 启动SAS, 认识界面。
2. 输入1.1.3的例子。在运行记录窗口查看有无错误。有错时回到程序窗口调回程序修改。
3. 打开SAS管理器查看各数据库的内容列表。
4. 启动SAS/INSIGHT, 打开SASUSER.GPA数据集。作各变量的直方图, 查看其分布情况并简答。把GPA数据集按性别排序, 同性别内按GPA分数由大到小排序。
5. 用INSIGHT数据窗口输入C9501数据集。
6. 研究GPA分数的分布。说明极端值情况。在纸上画出GPA的盒形图, 并说明如何解释。通过

直方图、盒形图、各统计量、分布检验结果简述GPA分布的特点。

7. 给男女生观测指定不同颜色。画GPA对HSM的散点图。画各数值型变量的散点图矩阵。画HSM、HSS、HSE的三维散点图。简述GPA数据集各变量间的直观的相互关系。

第二章 SAS语言与数据管理

SAS系统强大的数据管理能力、计算能力、分析能力依赖于作为其基础的SAS语言。SAS语言是一个专用的数据管理与分析语言，它的数据管理功能类似于数据库语言(如FoxPro)，但又添加了一般高级程序设计语言的许多成分(如分支、循环、数组)，以及专用于数据管理、统计计算的函数。SAS系统的数据管理、报表、图形、统计分析等功能都可以用SAS语言程序来调用，只要指定要完成的任务就可以由SAS系统按照预先设计好的程序去进行，所以SAS语言和FoxPro等属于第四代语言。

本章简单介绍SAS语言的基本成分与规则，SAS语言如何用来管理数据，SAS语言作为一个统计计算语言的用法。

2.1 SAS语言构成

2.1.1 SAS语句

SAS语言程序由数据步和过程步组成。数据步用来生成数据集、计算、整理数据, 过程步用来对数据进行分析、报告。SAS语言的基本单位是语句, 每个SAS语句一般由一个关键字(如DATA, PROC, IN-

PUT, CARDS, BY)开头, 包含SAS名字、特殊字符、运算符等, 以分号结束。

SAS关键字是用于SAS语句开头的特殊单词, SAS语句除了赋值、累加、注释、空语句以外都以关键字开头。SAS名字在SAS程序中标识各种SAS成分, 如变量、数据集、数据库, 等等。SAS名字由1到8个字母、数字、下划线组成, 第一个字符必须是字母或下划线。SAS关键字和SAS名字都不分大小写。

2.1.2 SAS表达式

SAS数据步程序中的计算用表达式完成。表达式把常量、变量、函数调用用运算符、括号连接起来得到一个计算结果。

SAS常量主要有数值型、字符型两种, 并且还提供了用于表达日期、时间的数据类型。例如

- 数值型：12, -7.5, 2.5E-10
- 字符型：'Beijing', "Li Ming", "李明"
- 日期型：'13JUL1998'd
- 时间型：'14:20't
- 日期时间型：'13JUL1998:14:20:32'dt

数值型常数可以用整数、定点实数、科学计数法实数表示。字符型常数为两边用单撇号或两边用双撇号包围的若干字符。日期型常数是在表示日期的字符串后加一个字母d(大小写均可), 中间没有空

格。时间型常数是在表示时间的字符串后加一个字母t。日期时间型常数在表示日期时间的字符串后加字母dt。

因为SAS是一种数据处理语言, 而实际数据中经常会遇到缺失值, 比如没有观测到数值, 被访问人不肯回答, 等等。SAS中用一个单独的小数点来表示缺失值常量。

SAS变量的基本类型有两种：数值型和字符型。日期、时间等变量存为数值型。SAS的数值型变量可以存储任意整数、定点实数、浮点实数, 一般不关心其区别。数值型变量在数据集中的存贮一般使用8个字节。SAS的字符型变量缺省的长度是8个字符, 但是如果在INPUT语句中输入字符型变量时指定了长度则不受此限制。可以用LENGTH语句直接指定变量长度, LENGTH语句一般应出现在变量定义之前, 格式为:

LENGTH *变量名* $ *长度*;

例如

```
LENGTH  name  $  20;
```

SAS运算符包括算术、比较、逻辑等运算符。

算术运算符为+ - * / **, 运算优先级按通常的优先规则。

比较运算符用于比较常量、变量的值大小、相等, 包括

= ^= > < >= <= IN

EQ NE GT LT GE LE

其中EQ等名字和＝等特殊字符是同一运算符的等价写法。比较运算符得到“真”或“假”的结果,

主要用于需要条件的分支、循环等语句中。运算符IN是一个SAS特有的比较运算符, 用来检查某个变量的取值是否在一个给定列表中, 比如

```
prov in ('北京', '天津', '上海', '重庆')
```

可以判断变量prov的取值是否为四个直辖市之一。

逻辑运算符用来连接比较得到的结果以构成复杂的条件, 有三种逻辑运算符:

& (AND)　| (OR)　^ (NOT)

其中AND是& (与)的等价写法, OR是|(或)的等价写法, NOT是^ (非)的等价写法。例如

(salary >= 1000) AND (salary < 2000)
　表示工资收入在1000－2000之间(不含2000)
(age <= 3) OR (sex = '女')
　表示三岁以下(含三岁)的婴儿及妇女
NOT ((salary >= 1000) AND (salary < 2000))
　表示工资收入不在1000－2000之间

复杂的逻辑表达式最好用括号表示其运算优先级以免误记优先规则并可利于阅读程序。

其它的运算符还有用于连接两个字符串的|| (两个连续的|号), 用于取两个运算值中较大一个的<>(比如3 <> 5结果为5), 用于取两个运算值中较小一个的>< (比如3 >< 5结果为3)。注意<>符在有些语言中用作“不等于”比较算符, 而SAS中用法则较特殊。

2.1.3 SAS程序规则

SAS程序由语句构成。每个语句以分号结尾(最常见的SAS编程错误就是丢失分号)。因为分号作为语句结束标志, 所以SAS语句不需要单独占一行, 一个语句可以写到多行(不需任何续行标志), 也可以在一行连续写几个语句。SAS语言中只要允许用一个空格的地方就可以加入任意多个空白(空格、制表符、回车), 允许用空格的地方是名字周围、运算符周围。比如, 程序

```
proc print
        data=c9501;
    by              avg;

run;
```

和

```
proc print data=c9501;by avg;run;
```

是等效的。另外, SAS关键字和名字大小写不分, 但字符型数据值要区分大小写, 比如"Beijing" 和"BEIJING" 被认为是不同的数据值。

在SAS程序中可以加入注释, 注释使用C语言语法, 用/*和*/在两端界定注释, 这种注释可以出现在任何允许加入空格的位置, 可以占多行。我们一般只把注释单独占一行或若干行, 不把注释与程序代码放在同一行。注释的另一个作用是把某些代码暂时屏蔽使其不能运行。下面是一个注释的例子:

```
/* 生成95级1班考试成绩的数据集 */
data c9501;
……
```

SAS程序包括数据步和过程步两种结构, 每一个步是一段相对完整的可以单独运行的程序。数据步用来生成、整理数据和自编程计算, 过程步调用SAS已编好的处理过程对数据进行处理。自己用SAS编程序进行计算主要在数据步中进行。

SAS数据步以DATA语句开头, 以RUN语句结尾。DATA步中可以使用INPUT, CARDS, INFILE, SET, MERGE 等语句指定数据来源输入数据, 也可以用赋值、分支、循环等编程结构直接生成数据或对输入的数据进行修改。

2.2 SAS用作一般高级语言

SAS是一种专用的数据处理、统计计算语言, 但是它也包含一般的高级语言编程能力并扩充了许多数学、统计等方面的函数。我们先介绍SAS语言用来进行一般编程计算的功能, 然后再讲解其独特的数据处理功能。SAS数据步的数据输入、整理功能很强, 希望进行复杂的数据管理的读者可以根据本节和下节的内容用SAS实现强大的数据管理功能。希望实现自己的统计计算算法的读者可以用SAS数据步编程, 但是我们建议使用S来编算法程序, 因为用S编程更方便。

SAS语言的编程计算能力主要由SAS数据步提供(另外SAS还提供了一个SAS/IML模块可以进行向量、矩阵运算, 读者有兴趣可以自己学习)。所以, 下面给出的例子如果没有写DATA语句实际应该在例子前面加上DATA语句, 在后面加上RUN语句才能运

行。DATA语句以关键字DATA开头, 后面给出一个数据集名, 这是本数据步要生成的数据集的名字, 例如:

```
data tmp1;
```

也可以省略数据集名, 这时SAS自动生成一个临时数据集名。还可以使用特殊名字_NULL_, 表示本数据步不生成数据集。

2.2.1 赋值语句

在SAS中用赋值语句计算一个值并存放到变量中。格式为

变量名＝表达式;

例如:

```
avg = (math + chinese/120*100)/2;
isfem = (sex='女');
y=sin(x)**2;
newv = .;
```

其中第一个赋值语句用一个公式计算平均分数。第二个生成一个取值为0或1的变量, 性别为女时为1, 否则为0。第三个使用了正弦函数和乘方运算。第四个给变量赋了缺失值。

注意想试验上述语句要把它们放入数据步中, 并且等号右边的表达式中的各变量应该是存在的, 否则会得到缺失值结果。

2.2.2 输出语句

SAS数据步的输出一般是数据集, 用赋值语句计算的结果会自动写入数据集。SAS也提供了一个PUT语句, 可以象其它语言程序的PRINT, WRITE(*, *), printf等语句一样立即显示输出结果。PUT语句在关键字后面列出要输出的各项, 每一项可以是变量名或字符串, 不能为数值常量或表达式, 各项之间用空格分开。PUT语句的输出结果显示在LOG窗口。例如:

```
data;
  x=0.5;
  y=sin(x);
  put  'Sine function value of '
       x  'is '  y;
run;
```

结果将在运行记录窗口显示一行

```
      Sine function value of 0.5 is 0.4794255386
```

另外, 在PUT语句中使用“变量名=”来指定输出项可以显示带有变量名的输出结果, 比如把上程序中的PUT语句改为

```
  put  x=  y=;
```

则结果在LOG窗口显示为

```
      X=0.5 Y=0.4794255386
```

PUT语句的输出项还可以指定具体列位置, 比如, 下面的PUT语句指定把X数值显示在第10–20列, 把Y数值显示在第30–40列, 并保留6位小数:

```
put x 10-20 .6 y 30-40 .6;
```

在指定的列位置内, 数值型数据靠右对齐, 字符型数据靠左对齐。要保留的小数位数写在一个小数点后面, 如果变量为整数值或者字符型则不必指定小数位数。

PUT语句还可以使用类似C、FORTRAN语言的“域宽.精度”方式指定输出的宽度和精度, 例如,

```
put x 20.8 y 20.8;
```

使X占用第1—20列, 8位小数, 右对齐; Y占用第21—40列, 8位小数, 右对齐。其中20.8是多种SAS输出格式中最常见的一种, 用于输出数值型数据, 小数点前面为输出宽度, 小数点后面为输出精度(小数位数)。对于字符型变量, 要指定其输出宽度可以用“$宽度.”的格式, 如“$10.”指定字符型变量输出宽度为10位。输出占不满指定宽度时, 数值型数据向右对齐, 字符型数据向左对齐。

字符型常量可以用于PUT语句中, 但不能规定占用的列位置或者宽度。

如果希望PUT语句的输出不产生换行, 使下一个PUT的结果可以显示在同一行, 只要在PUT语句结尾处加一个@符, 如:

```
put  i  @;
```

PUT语句的输出结果缺省情况下被送到运行记录窗口。在PUT语句之前用FILE语句可以改变PUT语句的输出目的地。比如, 在PUT语句之前用

```
file print;
```

可以把PUT语句的输出转向到输出窗口。在FILE语句中指定一个包含文件名的字符串可以把PUT语句的输出转向到此文件中。比如

```
file 'tmp.out';
```

把后续的PUT语句输出转向到当前工作目录下的文件“tmp.out”中, 生成输出文件tmp.out。注意当前工作目录在SAS状态栏的右方显示, 双击可以更改。文件名也可以指定全路径, 比如"C:\SAS\TMP.OUT"。

2.2.3 分支结构

如果需要在某条件满足时执行某一操作, 可以用

IF *条件* THEN *语句*;

的结构。比如, 如果X为正数则显示“X为正数”, 可以用

```
IF  x > 0  THEN  PUT  'X为正数';
```

有时我们在条件成立时需要进行的操作无法用一个语句完成, 这时可以使用SAS提供的**复合语句**功能: 只要把若干个语句用“DO;”语句和“END;”语句包围起来, 就可以把它们看作是一个语句, 就可以用在需要指定一个语句的地方。比如, 当X为正数时不仅显示X为正数, 而且将其加倍并显示值, 可以用如下带有复合语句的IF结构:

```
IF  x>0  THEN  DO;
    PUT  'X为正数';
    x = 2*x;
    PUT  x=;
    END;
```

以上的IF结构的用法只规定了条件成立时的操作, 如果同时需要规定条件不成立时进行什么操作, 使用带有ELSE字句的IF结构:

IF *条件* THEN *语句*;
ELSE *语句*;

其中“*语句*”均可以是复合语句。例如, 当X为非负时将X加倍, 为负时将X取绝对值, 用如下程序:

```
IF  x>=0  THEN  x=2*x;
ELSE  x = -x;
```

注意SAS的分支结构的写法与其它语言有些不同, 它不用ENDIF结束。

SAS的IF结构允许嵌套, 但SAS不提供IF－ELSEIF－ELSE的多分支结构。SAS的SELECT结构提供了更为灵活的多分支结构, 可以实现比其它语言的IF－ELSEIF－ELSE结构更强的功能。SELECT 结构有两种基本用法, 第一种为:

SELECT (*选择表达式*);
  WHEN(*值列表*) *语句*;
  WHEN(*值列表*) *语句*;
  ……
  OTHERWISE *语句*;
END;

其中“*选择表达式*”是一个取数值、字符型值的变量或表达式, “*值列表*”为一项或者若干项, 多项之间逗号分开, 每项可以是一个与选择表达式相同取值类型的表达式。“*语句*”可以是单个语句或复合语句。执行SELECT结构时, 先计算出选择表达式和值列表中的所有值, 然后把选择表达式值由前向后与值列表中的值相比, 发现相等值则执行对应的语句, 然后退出SELECT结构(不再查看后面的值列表)。如果选择表达式的值不等于任何值列表中的值则执行OTHERWISE对应的语句, 这种情况下没有OTHERWISE语句会出错。例如:

```
SELECT(month);
  WHEN('Feb', 'Mar', 'Apr')  put  '春天';
  WHEN('May', 'Jun', 'Jul')  put  '夏天';
  OTHERWISE  put  '秋天或冬天';
END;
```

根据MONTH值不同显示不同的季节。

SELECT语句的另一种形式为:

SELECT;
  WHEN(*条件*) *语句*;
  WHEN(*条件*) *语句*;
  ……
  OTHERWISE *语句*;
END;

这种SELECT语句没有选择表达式, 而是在每一个WHEN语句指定一个条件(逻辑表达式), 执行第一个满足条件的WHEN后的语句。如果所有条件都不满足则执行OTHERWISE后的语句。例如:

```
SELECT;
    WHEN(age<=12)  put  '少年';
    WHEN(age<35)  put  '青年';
    OTHERWISE  put  '中老年';
END;
```

注意上例中第二个WHEN语句的条件等价于age>12 and age<35, 因为如果年龄小于等于12的话则会执行第一个WHEN语句, 然后退出SELECT结构, 根本不会判断第二个条件。这与其它语言中的IF－ELSEIF－ELSE结构的用法是一致的。

2.2.4 循环结构

SAS数据步可以使用丰富的循环结构, 主要的是两种: 计数DO循环和当型、直到型循环。

计数DO循环的写法是:

DO *计数变量* = *起始值* TO *结束值* BY *步长*;
循环体语句……;
END;

在DO和END之间可以有多个语句。程序先把计数变量赋值为起始值, 如果此值小于等于结束值则执行循环体语句, 然后把计数变量加上步长, 再判断它是否小于等于结束值, 如果是则继续执行循环体, 直到计数变量的值大于结束值为止。上述结构中“BY 步长”可以省略, 这时步长为1。如果步长取负值, 则继续循环的条件是计数变量大于等于结束值。例如:

```
data;
  DO  i = 1  TO  20  BY  2;
    j = i**3;
    put  i  3.  j  5.;
  END;
run;
```

可以输出一个1, 3, 5, 7, …, 19的立方表。

在循环体中可以用LEAVE语句跳出循环, 相当于C语言的break语句。例如在上例中的循环体最后加上这样一句可以在立方大于1000时停止循环:

```
if  j>1000  then  LEAVE;
```

在循环体内用CONTINUE语句可以立即结束本轮循环并转入下一轮循环的判断与执行。比如:

```
data;
  do x=0 to 3.1415926  by  0.01;
```

```
    y = sin(x);
    if y<0 or y>0.5 then CONTINUE;
    z = cos(x);
    put  x  5.2  y  20.7  z  20.7;
  end;
run;
```

这个程序对0到π之间的数每隔0.01计算正弦值, 如果正弦值不在0到0.5之间则考虑下一个值, 否则计算余弦值并显示三个变量。

当型循环的语法是:

DO WHILE *循环继续条件*;
循环体语句……;
END;

程序先判断循环继续条件是否成立, 成立时执行循环体语句, 再判断循环继续条件, 如此重复, 直到循环继续条件不再成立。例如, 下面的程序判断1333333是不是素数:

```
data;
  x=1333333;
  i=3;
  DO  WHILE  (mod(x,i) ^= 0);
    i=i+2;
  END;
  if i<x then put x '不是素数';
  else  put  x  '是素数';
run;
```

其中mod(x,i)表示x除以i的余数。

直到型循环的写法是:

DO UNTIL *循环退出条件*;
循环体语句……;
END;

程序先执行循环体, 然后判断循环退出条件是否成立, 成立则结束循环, 否则继续。注意每轮循环都是先执行循环体再判断是否退出。例如:

```
data;
  n=0;
  do until (n>=5);
     n+1;
     put n=;
  end;
run;
```

可以依次输出n=1, 2, 3, 4, 5, 当n＝5时退出条件“n>=5”满足, 循环结束。上例中语句n+1是一种特殊的写法, 叫做**累加语句**, 等价于n=n+1。

事实上, SAS的循环语句比上面所述还要灵活得多, 它在DO语句中可以指定一个循环列表, 比如:

```
data;
  do i=3,7, 11 to 17 by 3 while (i**2<200);
     j=i**2;
     put i j;
  end;
run;
```

循环变量i取5, 7, 11, 14循环体被执行, 当i取17时i的平方为289故循环体不被执行, 循环结束。注意WHILE条件只作用于用逗号隔开的最后一项。

2.2.5 数组

SAS可以把一组同为数值型或同为字符型的变量合在一起, 使用同一个名字称呼, 用下标来区分。这与通常的程序设计语言中的数组略有区别, 通常的程序设计语言中数组元素没有对应的变量名, 而SAS数组每个元素都有自己的变量名。

一. 数值型数组

定义数值型数组的格式为:

ARRAY *数组名(维数说明) 数组元素名列表 (初始值表)*;

例如:

```
ARRAY  tests(3)  math chinese english  (0, 0, 0);
```

数组名是一个合法的SAS名字且不能与同一数据步中的变量重名。对一维数组, 维数说明只要说明元素个数, 这时下标从1开始。数组元素名列表列出这个数组的各个元素实际代表的变量名, 各变量名以空格分隔。比如, 上例中tests(1)代表数学成绩, tests(2)代表语文成绩, tests(3)代表英语成绩。初始值表给各数组元素赋初值, 按顺序对应。

数组说明中初始值表可以省略, 这时其初始值为相应数组元素的值(如果其数组元素还没有值则初值为缺失值)。

数组说明中的数组元素名列表可以省略, 这时其元素也有对应的变量名, 变量名为数组名后附加序号, 比如:

```
ARRAY  x(3);
```

中数组x的各元素名为x1, x2, x3。也可以在说明维数时用“下标下界:下标上界”来说明一个其它的下标下界, 如

```
ARRAY  sales(95:97)  yr95－yr97 ;
```

这 时sales(95) 为yr95, sales(96) 为yr96, sales(97) 为yr97。上面的变量名列表是一种特殊的语法, 在用到变量名列表时如果连续写几个前面字母相同, 后面是连续的序号的变量, 只要写出第一个和最后一个, 中间用减号连接。

一维数组的维数说明还可以是一个星号, 这时数组大小由提供的元素列表中的变量个数决定, 如上面的数组tests可以等价地说明为:

```
ARRAY  tests(*)  math chinese english  (0, 0, 0);
```

对这样的数组可以用函数DIM(数组名)来获得其长度。

可以定义二维数值型数组, 只要在维数说明中指定用逗号分开的两个下标界说明, 例如:

```
array  table(2,2)  x11 x12 x21 x22;
```

说明table(1,1)为x11, table(1,2)为x12, table(2,1)为x21, table(2,2)为x22。二维数组元素按行排列。

二. 字符型数组

定义字符型数组的语法略复杂, 它需要加一个$符来说明数组元素类型为字符型, 并且要说明每一元素所能存储的字符串的最大长度。说明格式如下:

ARRAY *数组名*(*维数说明*) $ 元素长度说明 *数组元素名列表* (*初始值表*);

例如:

```
ARRAY  names(3)  $ 10  child  father  mother;
```

字符型数组其它方面用法与数值型相同。

三、临时数组

上面格式说明的数组都是把若干个变量集合在一起使用同一个数组名称呼, 每个数组元素是一个独立的变量。SAS也提供了与其它程序设计语言相同的数组, 即数组元素只由数组名和序号决定, 没有对应的变量名。这种数组叫做临时数组, 定义格式为:

ARRAY *数组名*(*维数说明*) _TEMPORARY_ (*初始值表*);

可见临时数组就是在数组说明中用_TEMPORARY_代替了数组元素列表。例如:

```
ARRAY  x(3)  _TEMPORARY_  (0, 0, 0);
```

说明了一个有三个元素的临时数组x。其元素为x(1), x(2), x(3), 即使变量x1, x2, x3存在也与此数组无关。

临时数组的特点是它只用于中间计算, 最终不被写入数据集。并且临时数组与其它变量不同的是, 它在数据步隐含循环(后面会解释此概念)中能自动保留上一步得到的值。临时数组当然也可以有多维数组, 或字符型数组。

四、使用数组

临时数组的使用与其它程序设计语言中的数组作用相同, 可以存放性质类似的数据进行处理。SAS以变量为元素的数组可以方便变量的循环处理。比如, 读入了comp1-comp10十个计算机销售额变量, prin1-prin6六个打印机销售额变量, 希望计算其总和, 可以用如下的数组说明与DO循环配合进行:

```
data sales;
  input  comp1-comp10  prin1-prin6;
  ARRAY  y(*) comp1-comp10  prin1-prin6;
  tot=0;
  do i=1 to DIM(y);
    tot + y(i);
  end;
  cards;
.........
;
run;
```

此例中数组说明用了星号说明维数, 求总和时用了累加语句。事实上, 在数组说明的数组元素列表部分除了列出具体的变量名表外, 还可以用特殊名字_NUMERIC_代表所有数值型变量的列表, 用_CHARACTER_代表所有字符型变量的列表, 用_ALL_代表所有变量的列表(因为数组元素必须是

相同类型所以用_ALL_时所有变量应该同为数值型或同为字符型, 否则出错)。所以上例中的数组y的说明中还可以用_NUMERIC_或_ALL_代替变量名列表。

实际上, SAS为变量累加提供了一个专门的函数SUM(OF ···), 比如上面的tot变量可以用SUM(OF comp1-comp12 prin1-prin6)计算。这个例子为了说明如何循环处理多个变量所以不用SUM函数。

2.2.6 函数

SAS提供了比一般程序设计语言多几倍的标准函数可以直接用在数据步的计算中, 其中包括所有语言都有的数学函数、字符串函数, 还包括特有的统计分布函数、分位数函数、随机数函数、日期时间函数、财政金融函数, 等等。

这些函数的调用方法类似其它语言, 比如求x1, x2, x3三个自变量的和可以用函数SUM(x1,x2,x3)。另外, SAS还提供了函数调用的另一种语法以便于把多个数据集变量作为函数自变量, 其格式为“函数名(OF 变量名列表)”, 其中变量名列表可以是任何合法的变量名列表, 比如x1, x2, x3的和等价地可以用SUM(OF x1 x2 x3)或SUM(OF x1-x3)表示。注意两种写法不能混在一起, 比如SUM(OF x1,x2,x3)和SUM(x1-x3)都是错的。

本小节对重要的函数加以介绍, 其它详见《SAS软件: Base SAS软件使用手册》(高惠璇等编译, 中国统计出版社出版)。

一、数学函数

- ABS(x) 求x的绝对值。
- MAX(x1,x2,⋯,xn) 求所有自变量中的最大一个。
- MIN(x1,x2,⋯,xn) 求所有自变量中的最小一个。
- MOD(x,y) 求x除以y的余数。
- SQRT(x) 求x的平方根。
- ROUND(x,eps) 求x按照eps指定的精度四舍五入后的结果, 比如 ROUND (5654.5654, 0.01) 结果为5654.57, ROUND(5654.5654,10)结果为5650。
- CEIL(x) 求大于等于x的最小整数。当x为整数时就是x本身, 否则为x右边最近的整数。
- FLOOR(x) 求小于等于x的最大整数。当x为整数时就是x本身, 否则为x左边最近的整数。
- INT(x) 求x扔掉小数部分后的结果。
- FUZZ(x) 当x与其四舍五入整数值相差小于1E−12时取四舍五入。
- LOG(x) 求x的自然对数。
- LOG10(x) 求x的常用对数。
- EXP(x) 指数函数e^x。
- SIN(x), COS(x), TAN(x) 求x的正弦、余弦、正切函数。

- ARSIN(y) 计算函数y=sin(x)在$x \in [-\frac{\pi}{2}, \frac{\pi}{2}]$区间的反函数, y取[−1,1]间值。
- ARCOS(y) 计算函数y=cos(x)在$x \in [0, \pi]$的反函数, y取[−1,1]间值。
- ATAN(y) 计算函数y=tan(x)在$x \in (-\frac{\pi}{2}, \frac{\pi}{2})$的反函数, y取$(-\infty, \infty)$间值。
- SINH(x), COSH(x), TANH(x) 双曲正弦、余弦、正切。
- ERF(x) 误差函数$\frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$。
- GAMMA(x) 完全Γ函数$\int_0^\infty t^{x-1} e^{-t} dt$。

此外还有符号函数SIGN, Γ函数一阶导数函数DIGAMMA, 二阶导数函数TRIGAMMA, Γ的自然对数LGAMMA, 误差函数余函数ERFC, ORDINAL函数, AIRY函数, DAIRY函数, Bessel函数JBESSEL, 修正的Bessel函数IBESSEL, 等等。

二. 数组函数

数组函数计算数组的维数、上下界, 有利于写出可移植的程序。数组函数包括:

- DIM(x) 求数组x第一维的元素的个数(注意当下界为1时元素个数与上界相同, 否则元素个数不一定与上界相同)。
- DIMk(x) 求数组x第k维的元素的个数。如DIM2(x)计算二维数组x第二维长度。

- LBOUND(x) 求数组x第一维的下界。
- HBOUND(x) 求数组x第一维的上界。
- LBOUNDk(x) 求数组x第k维的下界。
- HBOUNDk(x) 求数组x第k维的上界。

三. 字符函数

较重要的字符函数有:

- TRIM(s) 返回去掉字符串s的尾随空格的结果。
- UPCASE(s) 把字符串s中所有小写字母转换为大写字母后的结果。
- LOWCASE(s) 把字符串s中所有大写字母转换为小写字母后的结果。
- INDEX(s,s1) 查找s1在s中出现的位置。找不到时返回0。
- RANK(s) 字符s的ASCII码值。
- BYTE(n) 第n个ASCII码值的对应字符。
- REPEAT(s,n) 字符表达式s重复n次。
- SUBSTR(s,p,n) 从字符串s中的第p个字符开始抽取n个字符长的子串
- TRANWRD(s,s1,s2) 从字符串s中把所有字符串s1替换成字符串s2后的结果。

其它字符函数还有COLLATE, COMPRESS, INDEXC, LEFT, LENGTH, REVERSE, RIGHT, SCAN, TRANSLATE, VERIFY, COMPBL, DEQUOTE, INDEXW, QUOTE, SOUNDEX, TRIMN, INDEXW, 等等。

四. 日期和时间函数

常用日期和时间函数有:

- MDY(m,d,yr) 生成yr年m月d日的SAS日期值
- YEAR(date) 由SAS日期值date得到年
- MONTH(date) 由SAS日期值date得到月
- DAY(date) 由SAS日期值date得到日
- WEEKDAY(date) 由SAS日期值date得到星期几
- QTR(date) 由SAS日期值date得到季度值
- HMS(h,m,s) 由小时h、分钟m、秒s生成SAS时间值
- DHMS(d,h,m,s) 由SAS日期值d、小时h、分钟m、秒s生成SAS日期时间值
- DATEPART(dt) 求SAS日期时间值dt的日期部分
- INTNX(interval,from,n) 计算从from开始经过n个间隔后的SAS日期。其中interval 可以取'YEAR', 'QTR', 'MONTH', 'WEEK', 'DAY' 等。比如,

INTNX('MONTH', '16Dec1997'd, 3)结果为1998年3月1日。注意它总是返回一个周期的开始值。

- INTCK(interval,from,to) 计算从日期from到日期to中间经过的interval 间隔的个数, 其中interval 取'MONTH' 等。比如, INTCK('YEAR', '31Dec1996'd, '1Jan1998'd) 计算1996年12月31日到1998 年1 月1 日经过的年间隔的个数, 结果得2, 尽管这两个日期之间实际只隔1年。

其它日期和时间函数还有DATE, TODAY, DATETIME, DATEJUL, JULDATE, HOUR, MINUTE, SECOND, TIME, TIMEPART等。详见《SAS 系统– Base SAS软件使用手册》、《SAS系统– SAS / ETS 软件使用手册》。

五. 分布密度函数、分布函数

作为一个统计计算语言, SAS提供了多种概率分布的有关函数。分布密度、概率、累积分布函数等可以通过几种统一的格式调用, 格式为

- 分布函数值= CDF('*分布*', x <, *参数表*>);
- 密度值= PDF('*分布*', x <, *参数表*>);
- 概率值= PMF('*分布*', x <, *参数表*>);
- 对数密度值= LOGPDF('*分布*', x <, *参数表*>);
- 对数概率值= LOGPMF('*分布*', x <, *参数表*>);

CDF计算由'*分布*'指定的分布的分布函数, PDF计算分布密度函数值, PMF计算离散分布的分布概率, LOGPDF 为PDF的自然对数, LOGPMF 为PMF 的自然对数。函数在自变量x处计算, "<, 参数表>"表示可选的参数表。

分布类型取值可以为: BERNOULLI, BETA, BINOMIAL, CAUCHY, CHISQUARED, EXPONENTIAL, F, GAMMA, GEOMETRIC, HYPERGEOMETRIC, LAPLACE, LOGISTIC, LOGNORMAL, NEGBINOMIAL, NORMAL 或GAUSSIAN, PARETO, POISSON, T, UNIFORM, WALD 或 IGAUSS, WEIBULL。可以只写前四个字母。

例如, PDF('NORMAL', 1.96)计算标准正态分布在1.96处的密度值(等于0.05844), CDF('NORMAL', 1.96)计算标准正态分布在1.96处的分布函数值(等于0.975)。PMF对连续型分布即PDF。

除了用上述统一的格式调用外, SAS还单独提供了常用的分布函数。

- PROBNORM(x) 标准正态分布函数
- PROBT(x,df<,nc>) 自由度为df的t分布函数。可选参数nc为非中心参数。
- PROBCHI(x,df<,nc>) 自由度为df的卡方分布函数。可选参数nc为非中心参数。
- PROBF(x,ndf,ddf<,nc>) F(ndf,ddf)分布的分布函数。可选参数nc为非中心参数。

- PROBBNML(p,n,m) 设随机变量Y服从二项分布B(n,p), 此函数计算Y ≤ m 的概率。
- POISSON(lambda,n) 参数为lambda的Poisson分布Y≤n的概率。
- PROBNEGB(p,n,m) 参数为(n,p)的负二项分布Y≤m的概率。
- PROBHYPR(N,K,n,x<,r>) 超几何分布的分布函数。设N个产品中有K个不合格品, 抽取n个样品, 其中不合格品数小于等于x的概率为此函数值。可选参数r是不匀率, 缺省为1, r代表抽到不合格品的概率是抽到合格品概率的多少倍。
- PROBBETA(x,a,b) 参数为(a,b)的Beta分布的分布函数。
- PROBGAM(x,a) 参数为a的Gamma分布的分布函数。
- PROBMC 计算多组均值的多重比较检验的概率值和临界值。
- PROBBNRM(x,y,r) 标准二元正态分布的分布函数, r为相关系数。

六. 分位数函数

分位数函数是概率分布函数的反函数。其自变量在0到1之间取值。分位数函数计算的是分布的左侧分位数。SAS提供了六种常见连续型分布的分位数函数。

- PROBIT(p) 标准正态分布左侧p分位数。结果在−5到5之间。
- TINV(p, df <,nc>) 自由度为df的t分布的左侧p分位数。可选参数nc为非中心参数。
- CINV(p,df <,nc>) 自由度为df的卡方分布的左侧p分位数。可选参数nc为非中心参数。
- FINV(p,ndf,ddf <,nc>) F(ndf,ddf)分布的左侧p分位数。可选参数nc为非中心参数。
- GAMINV(p,a) 参数为a的伽马分布的左侧p分位数。
- BETAINV(p,a,b) 参数为(a,b)的贝塔分布的左侧p分位数。

七. 随机数函数

SAS可以用来进行随机模拟。它提供了常见分布的伪随机数生成函数。

均匀分布随机数 有两个均匀分布随机数函数: UNIFORM(seed), seed必须是常数, 为0, 或5位、 6位、7位的奇数。RANUNI(seed), seed为小于$2^{31}-1$的任意常数。在同一个数据步中对同一个随机数函数的多次调用将得到不同的结果, 但不同数据步中从同一种子出发将得到相同的随机数序列。随机数种子如果取0或者负数则种子采用系统日期时间。

正态分布随机数 有两种, NORMAL(seed), seed为0, 或5位、6位、7位的奇数。RANNOR(seed), seed为任意数值常数。

指数分布随机数 RANEXP(seed), seed为任意数值, 产生参数为1的指数分布的随机数。参数为lambda的指数分布可以用RANEXP(seed)/lambda得到。

另外若Y=alpha−beta*LOG(RANEXP(seed)), 则Y为位置参数为alpha, 尺度参数为beta的极值分布。若Y=FLOOR(−RANEXP(seed)/LOG(p)), 那么Y是具有参数p的几何分布变量。

伽马分布随机数 RANGAM(seed, alpha), seed为任意数值常数, alpha>0, 得到参数为alpha的伽马分布。设X=RANGAM(seed, alpha), 则Y=beta*X 是形状参数为alpha, 尺度参数为beta的GAMMA分布随机数。如果alpha是整数, 则Y=2*X是自由度为2*alpha的卡方分布随机数。

如果alpha是正整数, 则Y=beta*X 是Erlang分布随机数, 为alpha个独立的均值为beta的指数分布变量的和。

如果Y1=RANGAM(seed,alpha), Y2=RANGAM(seed,beta), 则Y = Y1 / (Y1+Y2) 是参数为(alpha, beta) 的贝塔分布随机数。

三角分布随机数 RANTRI(seed,h), seed为任意数值常数, $0 < h < 1$。此分布在0到1取值, 密度在0到h之间为$2x/h$, 在h到1之间为$2(1-x)/(1-h)$。

柯西分布随机数 RANCAU(seed), seed为任意数值常数。产生位置参数为0, 尺度参数为1的标准柯西分布随机数。Y=alpha+beta*RANCAU(seed) 为位置参数为alpha, 尺度参数为beta的一般柯西分布随机数。

二项分布随机数 RANBIN(seed,n,p)产生参数为(n,p)的二项分布随机数, seed为任意数值。

泊松分布随机数 RANPOI(seed,lambda)产生参数为lambda>0的泊松分布随机数, seed为任意数值。

一般离散分布随机数 RANTBL(seed, p1, …, pn)生成取1, 2, …, n的概率分别为p1, …, pn的离散分布随机数。

八. 样本统计函数

样本统计函数把输入的自变量作为一组样本, 计算样本统计量。其调用格式为“函数名(自变量1, 自变量2, …, 自变量n)”或者“函数名(OF 变量名列表)”。比如SUM是求和函数, 如果要求x1, x2, x3的和, 可以用SUM(x1,x2,x3), 也可以用SUM(OF x1-x3)。这些样本统计函数只对自变量中的非缺失值进行计算, 比如求平均时把缺失值不计入内。

各样本统计函数为:

- MEAN 均值
- MAX 最大值
- MIN 最小值

- N 非缺失数据的个数
- NMISS 缺失数值的个数。
- SUM 求和
- VAR 方差
- STD 标准差
- STDERR 均值估计的标准误差, 用STD / SQRT(N) 计算。
- CV 变异系数
- RANGE 极差
- CSS 离差平方和
- USS 平方和
- SKEWNESS 偏度
- KURTOSIS 峰度

注意: 数据集的存储一般是每行为一个个体的观测值, 每列是个体的一个属性(变量), 所以统计一般应该对列进行, 而不是象这里对行进行, 把各变量作为一个样本的各个观测处理。这里提供的函数主要用于进行一些自编程的计算或观测数据在数据集内排列非标准的情况。

2.2.7 SAS/IML矩阵功能简介

SAS/IML是SAS提供的一个可以进行矩阵运算编程的工具, 详细使用请参见有关资料或系统帮助(Help - Extended Help - SAS System Help: Main Menu - Help for SAS Products - SAS/IML)。它可以用来进行交互的矩阵运算, 也可以编好一个程序再一起运行。程序可以使用分支、循环、模块化子程序等控制结构。数据步中的函数大都能在SAS/IML中使用, SAS/IML也提供了一些特有的函数。SAS/IML的一个方便之处是它可以直接读取SAS的数据集并把结果写成SAS数据集, 它也有存取外部文件的功能。

要交互运行SAS/IML, 只要在程序窗口输入

```
proc iml;
  reset print;
```

提交此程序, 就可以进入交互的SAS/IML运算状态。退出用QUIT语句。SAS / IML 中可以使用标量、行向量、列变量和矩阵, 可以使用字符型数据。变量取名规则遵循SAS语言的统一规定, 变量可以存储标量、向量和矩阵。

赋值用等号。例如:

```
sc = 15.25;
vh = {1 2};
vh1=5:9;
vv = {3, 4};
mat1 = {1 2 3,
        4 5 6};
mat2 = {"Li"  "Ming",
        "Zhang"  "Chong"};
```

我们注意赋值语句是一个SAS语句, 它以分号结尾。上面定义了标量sc, 行向量vh和vh1, 列向量vv, 两行两列的矩阵mat1, 字符型矩阵mat2。由上可见, 写矩阵常量时, 行的元素之间以空格分隔, 行之间以逗号分割。可以用“开始:结尾”的写法生成一个等差数列行向量。

矩阵之间可以用<、 = 等符号进行元素两两间的比较。要得到一个标量的结果, 可以用ALL()函数表示自变量的各元素均为真(非零), 用ANY()函数表示自变量的元素中至少一个为真。可以用&、|、^ 连接两个逻辑型矩阵(元素间的与、或、非)。

用||表示矩阵左右连接, 用//表示矩阵上下连接。**x‘**表示x的转置。

SAS/IML中可以进行通常的矩阵加减乘(+ 、 −、 *)运算, 也可以进行对应元素之间的乘除(# 、 /) 运算。矩阵加减必须两个矩阵大小完全相等, 或者其中一个是标量。矩阵乘法要求第一矩阵的列数等于第二矩阵的行数, 或其中一个是标量。矩阵元素之间的乘除运算是对应元素进行乘除, 当两个矩阵大小完全相同时可以进行运算, 其中有一个是标量时可以进行运算, 另外, 如果其中有一个是行(列)向量而其长度与另一个矩阵的列数(行数)相同也可以进行运算。矩阵逆要用INV()函数运算。

为了读入一个数据集, 先打开数据集, 用如

```
USE  sasuser.c9501;
```

然后用READ ALL VAR*变量列表*的格式读入数据集的各变量值, 例如:

```
READ ALL VAR {name math chinese};
print name math chinese;
```

这时三个变量都可以作为列向量来使用。

2.3 SAS语言的数据管理功能

前面说过, SAS语言是一种专用的数据管理、分析语言, 它提供了很强的数据操作能力。这些数据操作能力表现在它可以轻易地读入任意复杂格式的输入数据, 并可以对输入的数据进行计算、子集选择、更新、合并、拆分等操作。另外, SAS系统还提供了用来访问其它数据库系统如Sybase、Oracle的接口, 访问各种微机用数据库文件如FoxPro、Excel的接口及向导, 并提供了一个SQL过程来实现数据库查询语言SQL的功能。SAS 语言直接、间接用于数据管理的语句很多, 这里只能拣最常用的介绍。

2.3.1 SAS数据步的运行机制

SAS语言的自编程计算功能主要在数据步实现, 一个SAS数据步相当于一个单独运行的程序。但是, SAS语言又是一个专用数据处理语言, 所以SAS数据步有其它语言所没有的特点。我们以下面的简单例子说明这一点:

```
data a;
  put x= y= z=;
  input x y;
  z=x+y;
```

```
  put x= y= z=;
  cards;
10 20
100 200
;
run;
```

运行后在LOG窗口显示如下记录:

```
  ......

  X=. Y=. Z=.
  X=10 Y=20 Z=30
  X=. Y=. Z=.
  X=100 Y=200 Z=300
  X=. Y=. Z=.
  NOTE: The data set WORK.A has 2 observations and 3 variables.
  ......
```

这个程序的运行流程是这样的:

1. DATA语句标志了数据步开始, 并指定了数据步结束时要生成的数据集名字为A(实际是WORK.A)。

2. 第一个PUT语句要输出变量X、Y、Z的值但它们还都没有定义, 所以LOG窗口的结果显示为三个缺失值。

3. 下面是INPUT语句, 它从CARDS语句后面的数据行中读取变量X的值10, 变量Y的值20。

4. 下一个赋值语句计算变量Z的值得到30。因此, LOG中的第二行输出显示三个变量的值分别为10、20、30。

5. 从CARDS语句开始到空分号行的各行是非执行的, 程序运行到RUN语句, 发现这是本数据步的最后一个语句, 按一般的程序语言的规则, 程序到这里就应该结束了, 但是, SAS是一个专用数据处理语言, 如果按一般语言的规则, 程序中的第二行数据(100 200)就不能被读入。所以, 这个程序运行到RUN语句后, 把读入的观测(这是第一号观测)写入输出数据集, 然后:

6. 又返回到DATA语句后的第一个可执行语句开始执行, 并先把所有的变量置初值为缺失值。于是, 第一个PUT语句的结果显示三个变量均为缺失值, 而不是上一步的10、20、30。

7. 下一个INPUT语句从数据行中读入下一个观测, 把变量X、Y赋值100、 200。读取位置由运行时设置的一个数据指针指示。然后计算变量Z的值得300。于是PUT语句输出的X、Y、Z值分别为100、200、300。

8. 然后, 运行控制跳过CARDS语句到空语句, 到数据步结尾, 把第二号观测输出到数据集。

9. 再返回到数据步开头, 把变量值赋初值为缺失值, 所以第一个PUT语句输出的三个变量值为缺失值。

10. 然后运行到INPUT语句, 应该读入下一个观测, 但是查询数据指针发现已经读完了所有数据, 所以本数据步结束, 并把两个观测写入了数据集WORK.A中。

提交PROC PRINT;RUN;就可以显示此数据集的内容如下:

OBS	X	Y	Z
1	10	20	30
2	100	200	300

从这个例子可以看出SAS数据步程序和普通程序的一个重大区别: SAS 数据步如果有数据输入, 比如用INPUT、SET、MERGE、UPDATE、MODIFY 等语句读入数据, 则数据步中隐含了一个循环, 即数据步程序执行到最后一个语句后, 会返回到数据步内的第一个可执行语句开始继续执行, 直到读入数据语句(INPUT、SET、MERGE、UPDATE、MODIFY等)读入了数据结束标志为止才停止执行数据步, 并把读入的各个观测写入在DATA语句中指定的数据集。如果没有数据输入而只是直接计算, 则数据步程序不需要此隐含循环。数据步因为有这样一个隐含循环, 所以也提供了用来查询某一步是第几次循环的特殊变量_N_, 它的值为数据步循环计数值。数据步流程见图1。

2.3.2 用INPUT语句输入数据

在数据步中输入数据可以从原始数据输入, 也可以从已有数据集输入。从原始数据输入要使用INPUT语句来指定输入的变量和格式。数据行写在CARDS语句和一个只有一个顶头的分号的行之间。

最简单的INPUT语句使用自由格式: 按顺序列出每个观测的各个变量名, 中间用空格分开。变量如果

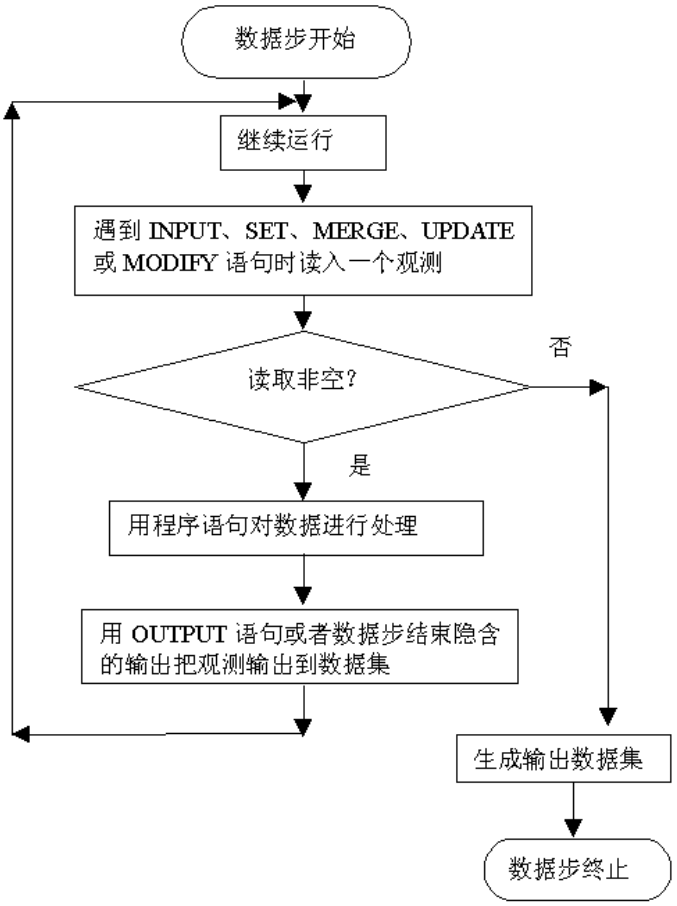


图 2.1: SAS 数据步流程图

是字符型的需要在变量名后面加一个$符号, $符与变量名可以直接相连也可以隔一个空格。例如:

```
data c9501;
  input name $ sex $ math chinese;
  cards;
李明 男 92 98
张红艺 女 89 106
王思明 男 86 90
张聪 男 98 109
刘颍 女 80 110
;
```

```
run;
```

注意这个例子的数据有五个观测, 四个变量, 每行数据的各变量之间用空格分隔。为输入这些数据, INPUT语句中依次列出了四个变量名, 并在字符型变量NAME和SEX后加了$符。要生成一个数据集这是最简单的写法。

使用自由格式也有一些限制条件, 如果不满足这些条件时需要改用其它输入格式:

- 数据每行为一个观测, 各数据值之间用空格或制表符分隔。
- 无论是字符型还是数值型缺失数据都必须用小数点表示。
- 字符型数据长度不能超过8个字符, 不允许完全是空白, 中间不允许有空白, 开头和结尾如果有空白将被忽略。
- 在INPUT语句中必须列出观测中的每一项数据对应的变量名而不能省略中间的某一个。

在满足以上条件时就可以使用自由格式, 它也有明显的优点: 使用简单；输入数据时不必上下对齐；不需要知道每个变量的具体列数而只需知道它的次序。

如果各数据行的各个数据项是上下对齐的, 还可以使用INPUT语句的列方式。这时, 除了在INPUT关键字后面列出变量名外, 还需要在每个变量名(及$符)后面列出该变量在数据行中所占据的

列起始位置与结束位置, 比如上面的例子可以改写成:

```
data c9501;
  input name $ 1-10 sex $ 11-13
        math 14-16 chinese 17-20;
  cards;
李明       男  92  98
张红艺     女  89 106
王思明     男  86  90
张聪       男  98 109
刘颍       女  80 110
;
run;
```

使用列方式时一定要正确数出每一项所占的位置。列方式有如下特点:

- 要求数据行各项上下对齐
- 各项之间可以没有任何分隔, 连续写在一起。
- 字符型数据长度可以超过8个字符, 中间可以有空格, 头尾的空格仍将被忽略。
- 不论字符型变量还是数值型变量如果指定列位置都是空白则输入值为缺失值。小数点仍表示数值型和字符型变量的缺失值。
- 可以只输入数据行中的某些项而忽略其它项。

列方式不要求数据项之间分开, 所以经常用来输入紧缩格式的数据。比如, 我们要输入一批身份证号码, 但只输入其中的出生年、月、日信息, 就可以用如下程序:

```
data pids;
  input year 7-10  mon 11-12  day 13-14;
  cards;
110103197512092232
110101196902150059
;
run;
```

列格式可以与自由格式混用, 见1.1.3的例子。

如果需要完全原样地输入字符型数据(包括头尾空格、单独的小数点), 可以用有格式输入, 即在字符型变量名和$符后加上一个输入格式如CHAR10.表示读入10个字符。

有特殊格式的数据需要用有格式输入, 即在变量名后加格式名。其中最常见的是用来输入日期。数据中的日期写法经常是多种多样的, 比如1998年10月9日可以写成“1998-10-9”, “19981009”, “9/10/98”等等, 为读入这样的日期数据就需要为它指定特殊的日期输入格式。另外, 日期数据在SAS中是按数值存储的, 所以如果要显示日期值, 也需要为它指定特殊的日期输出格式。例如:

```
data a;
  input date yymmdd8. sales;
  format date yymmdd10.;
  cards;
56-6-13    1100
67.12.15    1200
78 10 2    1300
891001     1400
19960101   1500
20020901   1600
;
```

```
run;
proc print;run;
```

其中日期数据占据8列位置, 如果不满8列要用空格补充, 不能让后面的数据进入这8列。这样可以输入没有世纪数, 年、月、日之间用减号、小数点、空格分隔的日期, 可以输入YYMMDD格式的六位数的日期(一位数的月、日前面补0), 可以输入带世纪数的YYYYMMDD格式的日期(一位数的月、日前面补0)。FORMAT语句规定输出日期变量时使用的显示格式。结果为:

```
      1     1956-06-13     1100
      2     1967-07-11     1200
      3     1978-10-02     1300
      4     1989-10-01     1400
      5     1996-01-01     1500
      6     2002-09-01     1600
```

用YYMMDD10.输入格式可以输入带世纪数的中间有分隔符或无分隔的日期, 如:

```
data b;
  input date yymmdd10. sales;
  format date yymmdd10.;
  cards;
56-6-13    1100
67.12.15    1200
78 10 2    1300
891001     1400
19960101   1500
20020901   1600
1956-6-13  1100
1967.12.15  1200
1978 10 2  1300
```

```
19891001   1400
19960101   1500
20020901   1600
;
run;
proc print;run;
```

如果日期变量不是第一个, 则它的前一项最好使用列格式并且指定结束列号为日期值的前一列, 或者前一项也使用指定输入格式的输入方法, 并且使前一项的输入域宽占满日期前的列。如果用自由格式则当前一项与日期数据之间间隔超过一个空格时有可能导致日期读入时对不准位置。如果数据是按列对齐的, 还可以在日期变量前加上“@开始列号”说明日期变量开始读取的位置, 比如:

```
data b;
  input sales @15  date yymmdd10. ;
  format date yymmdd10.;
  put date=;
  cards;
1100            56-6-13
1200            67.12.15
;
run;
```

数据中日期是从第15列开始的。如果在上面去掉“@15”, 则读入的DATE 变量为缺失值。但是如果把日期值与前一个数据值只隔开一个空格则可以用自由格式。

有格式读取还可以与自由格式结合起来使用, 在INPUT语句中指定“变量名: 格式”表示按位置读取当前第一个非空列开始的值并用指定的输入格式

转换, 比如, 中间有日期又没有对齐时就可以用下例这样的方法读取:

```
data b;
  input sales date : yymmdd10. ;
  format date yymmdd10.;
  put date=;
  cards;
1100    56-6-13
1200       67.12.15
;
run;
```

INPUT语句有十分复杂的使用方法, 可以处理几乎是任意复杂的数据输入问题, 这里我们就不详细讲了, 有兴趣的读者可以参考《SAS系统－Base SAS软件使用手册》。

2.3.3 变量属性

前面提到用LENGTH语句可以规定变量的长度。变量长度是数据集变量属性之一，变量的属性包括:

(1) 字符型还是数值型。INPUT语句读入字符型数据时要在变量名后面加$符。

(2) 变量标签(LABEL)。可以给变量加一个长度不超过40个字符的标签(可以用汉字，不超过20个汉字)，标签可以用在以后的报表中。

(3) 变量存储长度(LENGTH)。数值型数据一般长度为8字节，也可以对取值范围小的变量规定较小的长度以节省存储空间。字符型变量长度为其能存储的字符个数，缺省为8个字节。

(4) 变量的输出格式(FORMAT)。指定如何显示变量值。

(5) 变量的输入格式(INFORMAT)。指定如何把外部数据转换为SAS数据。

数据步中的ATTRIB语句可以指定变量的这些属性。格式为：

ATTRIB *变量名* 属性 *变量名* 属性 ···;

可以同时指定多个变量的属性。属性为“属性名=属性值”这样的写法，可以为一个变量同时指定多个属性。见如下的例子：

```
data sales;
  ATTRIB name LABEL="姓名" LENGTH=$10
         date LABEL="日期" FORMAT=yymmdd10.
              INFORMAT=mmddyy10.
         amount LABEL="金额" FORMAT=10.2;
  input name $ 1-10 date amount;
  cards;
张鹏       10/15/1998 2000
李志明     1/3/99  1500
王敏       11/5/99 3000
;
run;
proc print noobs label;
run;
```

此例的数据步中因为date已规定输入格式所以INPUT语句中可以直接输入。定义的变量标签被用于后面的PRINT过程中。

2.3.4 读入外部数据

文本格式的数据文件

对于小量的数据, 用CARDS语句和空语句把数据夹在中间放在数据步程序中就可以用INPUT语句输入数据。如果数据量很大(有时可以有上亿行、几千列), 直接把数据放在程序中不利于程序和数据的维护。这时, 一种办法是把原始数据放在一个普通的文本格式的文件中, 然后用INFILE语句指定输入文件名。例如, 我们可以把1.1.3中的数据行单独生成一个文本文件stud.txt, 假设放在了D:\USERS\SAS中, 可以用如下程序读入文件中的数据并生成数据集:

```
data c9501;
  infile 'd:\users\sas\stud.txt';
  input name $ 1-10 sex $ math chinese;
run;
proc print;run;
```

注意INFILE语句要写在INPUT语句之前, 有INFILE语句就不再有CARDS 语句和空语句。INFILE 关键字后面跟的是一个包含文件名的字符串, 可以使用全路径名, 如果只有文件名则在当前工作目录寻找。

微机格式的数据文件

SAS还可以读入其它格式的文件, 比如FoxPro、Excel等微机格式数据文件。这样的读入不必编程, 而可以使用SAS系统File菜单中的Import命令完成。虽然这不是SAS语言的内容, 但因为实际工

作中我们经常会需要读入这样的数据文件, 所以我们在这里加以简单介绍。

新版SAS提供了一个用于把其他格式的数据文件转换为SAS数据集的导入向导(import wizard)。在SAS/Base的支持下可以转换用分隔符分隔的文件, 用逗号分隔的文件, 用制表键分隔的文件, 用户自定义的格式。在SAS/ACCESS关于PC文件的接口的支持下可以转换dBase, Excel, Lotus文件等特殊格式的文件。导入向导可以从程序编辑窗口的文件菜单启动。例如, 我们有一个Excel文件在D:\USERS\SAS\C9501.XLS中(见图2.2):

	A	B	C	D
1	NAME	SEX	MATH	CHINESE
2	李明	男	92.00	98.00
3	张红艺	女	89.00	106.00
4	王思明	男	86.00	90.00
5	张聪	男	98.00	109.00
6	刘颖	女	80.00	110.00
7				

图 2.2: Excel文件

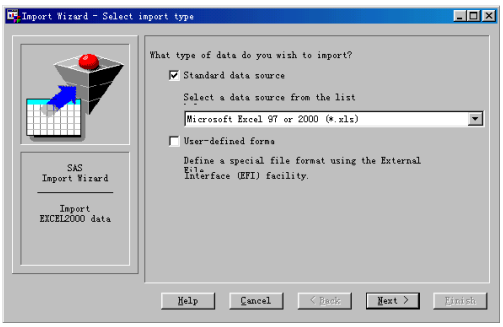


图 2.3: 导入向导–选择文件类型

为了由它转换得到SAS数据集, 在程序窗口中启动File菜单, 选“Import”, 这时出现一个选择文件类型的画面(图2.3): 选中标准文件格式(Standard data source), 并单击向下箭头打开一个下拉列表, 从中选“Microsoft Excel 97 or 2000 Spreadsheets(*.xls)”, 按Next钮继续, 出现一个选择文件名的画面, 可以在文本框中直接输入Excel文件的全路径名, 或按Browse钮从目录中选取文件。继续后出现选择目标位置的画面, 这是要求输入一个结果数据集的名

字和数据库位置, 数据库已选WORK我们可以不变, 在数据集名处输入C9501A, 按Finish钮可以生成数据集WORK.C9501A。如果在最后一步不按Finish钮而是按Next钮可以提示保存导入用的IMPORT过程的程序。例如本例得到的程序为：

```
PROC IMPORT OUT= WORK.c9501a
            DATAFILE= "D:\users\sas\c9501.xls"
            DBMS=EXCEL2000 REPLACE;
     GETNAMES=YES;
RUN;
```

这样我们下一次需要转换Excel数据时就可以直接修改以上程序而不需要再使用导入向导。在重复单一的工作时编程序要比使用菜单界面方便得多。

与大型数据库的接口

SAS提供了两种办法可以访问大型数据库。SAS / ACCESS 可以直接连接Oracle、Sybase、SQL Server等大型数据库。为了访问储存在这些数据库中的表, 需要对数据库中的表在SAS中建立访问描述文件(access descriptor), 和视图描述文件(view descriptor)。例如, 在数据库服务器DBIN中有一个数据库Finance, 其中有一个表Sales, 用户名guest用密码anyone可以访问此库, 就可以用以下程序在SAS中建立访问描述文件和视图文件:

```
PROC ACCESS DBMS=SYBASE;
    CREATE sasuser.sales.ACCESS;
    SERVER='DBIN';
    DATABASE='Finance';
    TABLE='Sales';
```

```
    USER='guest';
    PASSWORD='anyone';
    CREATE sasuser.salesall.VIEW;
    SELECT ALL;
RUN;
```

其中大写的部分是固定的。这段程序首先生成了访问描述文件SASUSER. SALES. ACCESS, 然后由此访问描述文件生成了视图文件SASUSER. SALESALL. VIEW。在SAS中视图文件和数据集的使用是一样的, 可以使用数据集的地方都可以使用视图文件。

对于SAS没有直接支持的数据库管理系统, 我们在MS Windows下总可以使用ODBC数据库接口来连接到数据库, 这要求我们在安装SAS时安装了ODBC驱动。为了访问外部数据库, 首先要在计算机上安装该数据库的客户端驱动程序, 然后在Windows的控制面板中打开ODBC的控制, 新建一项ODBC数据源, 输入该数据库管理器的名字, 数据库名。以我们刚刚用过的数据库为例，假设建立了ODBC数据源finodb, 就可以使用PROC SQL程序来建立视图:

```
PROC SQL;
    CONNECT TO ODBC (DSN='finodb'
        UID='guest'  PWD='anyone');
    CREATE VIEW sasuser.sales2 AS
        SELECT * FROM CONNECTION TO ODBC (
             SELECT * FROM Sales );
QUIT;
```

其中的CONNECT TO语句用来建立到数据库的连接, 后面的CREATE VIEW语句建立一个视图, 但是

图的数据来源是ODBC数据源, 所以AS后面是关键字SELECT * FROM CONNECTION TO ODBC, 然后在括号中给出了具体的从外部数据库中取得一个查询结果的SQL语句。这个SQL语句可以用来取表的一个子集构成视图。生成的SASUSER.SALES2是一个SAS视图, 在访问时将用保存的连接参数临时去访问外部数据库来得到数据。

这种使用PROC SQL直接连接外部数据库的方法也适用于非ODBC的数据源。

2.3.5 数据集的复制与修改

前面讲述了如何从原始数据生成SAS数据集。我们还可以用SET语句把一个已有数据集复制到一个新数据集, 同时还可以进行修改。如果只是复制数据集，也可以用SAS管理器(SAS Explorer)完成。

比如要把数据集WORK.C9501复制为数据集SASUSER.CLS, 只要用如下程序:

```
data sasuser.cls;
  set c9501;
run;
```

这样的程序流程中也有一个隐含循环, SET是读取观测的语句，程序在数据步内反复循环, 直到输入数据集C9501 最后一个观测读过。

在复制的同时我们还可以用SAS程序语句对生成的数据集进行修改。比如, 我们把超过100分的语文成绩都改为100分, 就可以用如下程序:

```
data c9501a;
  set c9501;
```

```
  if chinese>100 then chinese=100;
run;
```

当然, 这种修改也可以在读入原始数据的数据步中使用而不限于使用SET的数据步。也可以生成新的变量。

在数据步中可以用KEEP语句或DROP语句指定要保留的变量或要丢弃的变量。比如,

```
data c9501b;
  set c9501;
  keep name avg;
run;
```

生成的数据集C9501B只包含NAME和AVG两个变量。用KEEP语句指定要保留的变量。用DROP语句指定要丢弃的变量, 比如上例中的KEEP语句可以换成:

```
  drop sex math chinese;
```

用这种方法可以取出数据集的一部分列组成的子集。

也可以指定一个条件取出数据集的某些行组成的子集。比如, 我们希望取出数学分数90分以上, 语文分数100分以上的学生的观测, 可以用如下的“子集IF语句”:

```
data c9501c;
  set c9501;
  IF math>=90 and chinese>=100;
run;
```

注意子集IF语句不同于我们前面所讲的分支语句, 它没有THEN部分, 只有条件, 用于取出满足条件的行子集。

在用SET语句引入数据集时可以给引入的数据集加选项，选项放在数据集名后的括号内：

数据集名 (*数据集选项*)

选项包括KEEP=, 表示引入时只要指定的变量; DROP=, 表示不引入指定的变量；OBS=, 表示读取观测时读到指定的序号为止(是序号而不是观测数); FIRSTOBS=, 表示从指定序号的观测开始读取而跳过之前的观测不读。例如下面的程序生成了一个巨大的数据集HUGE 然后用数据步从中复制了其前100行和前两个变量：

```
data huge;
  array x(10);
  do i=1 to 1000000;
     do j=1 to 10;
           x(j) = normal(0);
         end;
         output;
  end;
  drop i j;
run;

data new;
  set huge(obs=100 keep=X1 X2);
run;
```

2.3.6 用SET和OUTPUT语句拆分数据集

有时我们需要根据某一分类原则把数据行分别存放到不同的数据集。比如, 我们希望把数据集C9501中

的所有男生的观测放到数据集C9501M中, 把所有女生的观测放到C9501F中, 可以使用如下程序:

```
data c9501m c9501f;
  set c9501;
  select(sex);
    when('男') output c9501m;
    when('女') output c9501f;
    otherwise put sex= '有错';
  end;
  drop sex;
run;
proc print data=c9501m;run;
proc print data=c9501f;run;
```

这个程序中有两个地方需要注意: 在DATA语句中, 我们指定了两个数据集名, 这表示要生成两个数据集。程序中用SET语句引入了一个数据集, 这个数据集的观测如何分配到两个结果数据集中呢？关键在于OUTPUT语句。OUTPUT语句是一个可执行语句, 它使得当前观测被写到语句指定的数据集中。这样, 我们根据SELECT的结果把不同性别分别放到了两个不同数据集中。

OUTPUT语句还可以用来强行写入数据集而不必象我们在数据步流程图中说明的那样等到数据步最后一个语句完成。数据步中有了OUTPUT语句后数据步流程中不再有自动写入观测的操作, 而只能由OUTPUT语句指定输出。不指定数据集名的OUTPUT语句输出到第一个结果数据集。比如下面的程序生成一个包含1到10的及其平方的有10个观测的数据集:

```
data sq;
  do i=1 to 10;
```

```
    j=i*i;
    output;
  end;
run;
proc print;run;
```

如果删去上面的OUTPUT语句则结果数据集中只有i=11, j=100的一个观测。

2.3.7 数据集的纵向合并

几个结构相同的数据集可以上下地连接到一起。比如, 我们有四个班的学生情况的数据集Class1-Class4, 每个数据集包含一个班学生的学号、姓名、性别信息, 我们希望把这些数据集合并为一个大数据集, 可以用如下代码:

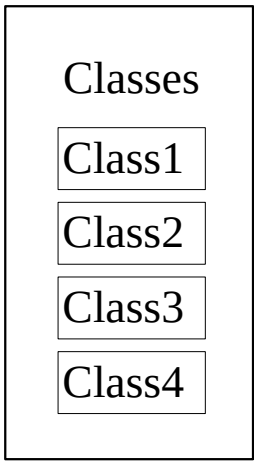


图 2.4: 纵向合并

```
data classes;
  set class1 class2
      class3 class4;
run;
```

可见, 要把若干个结构相同的数据集合并为一个数据集, 只要在DATA语句中指定要生成的大数据集的名字, 然后在数据步中使用SET语句并在SET语句中依次列出各小数据集。

有时我们需要在合并数据集时加入一个变量来指示每一个观测原来来自哪一个小数据集, 这可以在SET语句的每一个数据集名后面加一个括号,

里面写上IN=变量名, 变量名所给的变量取1表示观测来自此数据集, 取0表示观测非来自此数据集。例如, 在2.3.6中我们把C9501数据集按男、女拆分成了C9501M 和C9501F 两个数据集并抛弃了性别变量, 就可以用如下程序连接两个数据集并恢复性别信息:

```
data new;
  set c9501m(in=male) c9501f(in=female);
  if male=1 then sex='男';
  if female=1 then sex='女';
run;
```

在数据步中, 如果观测来自C9501M, 则变量MALE值为1, 如果观测来自C9501F 则变量FEMALE 值为1, 可以使用这两个变量的值定义新变量SEX。用数据集选项的IN=指定的变量不能直接进入结果数据集而只能用于数据步程序中。

2.3.8 数据集的横向合并

两个(或多个)数据集如果包含了同样的一些观测的不同属性(变量), 比如, 数据集C9501U包含学生的姓名、性别, 数据集C9501V包含学生的数学成绩, 数据集C9501W包含学生的语文成绩, 且各数据集的观测是按顺序一一对应的, 就可以用如下带有MERGE语句的数据步把它们左右横向合并到一个数据集NEW:

```
data new;
  merge c9501u c9501v c9501w;
run;
```

这样虽然可以横向合并数据集, 但是如果各数据集的观测顺序并不一样, 就会把不同人的成绩合并

到一起。所以横向合并一般应该采用按关键字合并的办法, 即先把每个数据集按照相同的、能唯一区分各观测的一个(或几个)变量排序, 然后用BY语句和MERGE语句联合使用, 这样即使原来观测顺序不一致或个数不同也能保证横向合并的结果正确。下例先把C9501数据集横向拆分为包含姓名、性别的数据集C9501X和包含姓名、数学成绩、语文成绩的数据集C9501Y, 然后按关键字横向合并:

```
data c9501x;
  set c9501;
  keep name sex;
run;
data c9501y;
  set c9501;
  keep name math chinese;
run;

proc sort data=c9501x;
  by name;
run;
proc sort data=c9501y;
  by name;
run;
data new;
  merge c9501x c9501y;
  by name;
run;
proc print;run;
```

其中的PROC SORT是排序过程, 用来把数据集按照某个变量的次序排序(这里是按变量NAME的次序排列, 用BY语句指定排序的变量名)。

2.3.9 用UPDATE语句更新数据集

如果我们发现数据集中的某些数据值有错误或者现在的值已经改变了, 我们可以从更正了的原始数据重新生成数据集, 或者使用更有效的方法, 即建立一个只包含新数据值的数据集, 用此数据集修改原数据集。使用如下的DATA步中可以实现数据集的更新:

DATA *新数据集名*;
  UPDATE *原数据集 更新用数据集*;
  BY 关键变量;
RUN;

例如, 比如我们发现数据集C9501中王思明的语文成绩实际应该是91分, 张红艺性别应为男, 可以先生成如下的只包含更正数据值的数据集, 不需要改的观测不列入, 不需要改的变量不列入或取缺失值:

```
data upd;
  input name $ sex $ chinese;
  cards;
张红艺 男 .
王思明 . 91
;
run;
```

然后, 把原数据集C9501和更新用数据集UPD均按姓名(NAME)排序:

```
proc sort data=c9501;
  by name;
run;
proc sort data=upd;
```

```
  by name;
run;
```

最后用UPDATE和BY更新得到新数据集NEW, 其中王思明的语文成绩改成了91分, 张红艺性别改成了男。

```
data new;
  update c9501 upd;
  by name;
run;
proc print;run;
```

但是, 这个新数据集中有一个错误: 王思明的语文成绩修改以后他的平均分也应作相应改动。所以此例应改为:

```
data new;
  update c9501 upd(in=in_upd);
  if in_upd=1 then
     avg = math*0.5 + chinese/120*100*0.5;
  by name;
run;
proc print;run;
```

2.3.10 SAS宏介绍

前面我们提到, SAS自行编程计算不太方便。其中的一个表现是, SAS数据步是单独执行的, 不能定义成一个模块化的可以带输入参数的子程序。这样, 我们在进行某些重复性工作时会感到不便。SAS特别提供了一种宏语言部分解决了这个问题。

一. 实例

我们先以一个实例来说明宏的作用。假设我们在工作目录下放了10个文件, 分别命名为df1.txt, df2.txt, ..., df10.txt。它代表了10个推销员的销售业绩, 每个文件的格式相同, 都包括日期、销售额。

为了把这10个推销员的记录都读入到SAS中，分别生成10个数据集然后把它们合并为一个大数据集，我们用了如下程序:

```
%MACRO SALREAD;
%DO NP=1 %TO 10;
  %let ff="df&NP..txt";
  %let fd=df&NP;
  data &fd;
    infile &ff;
          input date yymmdd10. sales;
          persid=&NP;
  run;
%END;

%LET setstm=SET;
%DO NP=1 %TO 10;
  %LET setstm=&setstm df&NP;
%END;
%PUT setstm;
data whole;
  &setstm;
run;

%MEND SALREAD;

%SALREAD;
```

这个程序不太好懂。我们从比较基本的概念讲起。

二. 宏变量

宏变量是比较基本的SAS 宏。用“%LET *宏变量名* = 值”的格式可以定义一个宏变量，用“&*宏变量名*.”的格式可以在任何地方带入宏变量的值。宏变量代换完全是一种字符串的替换，学过C语言的读者可以发现这与C中的宏定义是同一类型。例如:

```
%LET fd=df3;
%LET ff="df3.txt";
data &fd.;
  infile &ff.;
  input date yymmdd10. sales;
  persid=3;
run;
```

这里首先定义了两个宏变量ff和fd。注意ff的值中包括”df3.txt”两边的双撇号，但不包括表示%LET语句结束的分号；fd的值就是字符串df3。下面用“&fd.”引用了fd的值，用“&ff.”引用了ff的值，因此程序经过宏替换后实际上变成了:

```
data df3;
  infile "df3.txt";
        input date yymmdd10. sales;
        persid=3;
run;
```

定义宏变量时可以引用已定义的宏变量值。例如，对fd和ff的定义还可以这样写:

```
%LET fd=df3;
%LET ff="&fd..txt";
```

效果与原来写法相同。注意这里引用fd用了一个句点然后产生文件名有一个句点所以程序中fd后面有两个句点。

如果要显示宏变量的值，可以用%PUT宏命令，结果将显示在LOG窗口，如:

```
%PUT &fd.;
```

以“&宏变量名.”格式引用宏变量值时如果不引起混淆可以省略其中的句点。例如:

```
%PUT &fd;
```

三. 宏

宏变量一般只保存较短的字符串。为了实现较长的SAS程序的替换，可以定义一个SAS宏。SAS宏有点象是子程序，但是一定注意其纯字符串替换的特点。

简单的宏的定义方法如下:

%MACRO *宏名*;
...(定义宏的内容)
%MACRO *宏名*;

例如，下面的程序定义了一段用来列表的程序:

```
%MACRO topr;
proc print data=&thedat. noobs label;
  var &varlist.;
run;
%MEND topr;
```

然后我们定义了数据集名thedat和变量列表varlist后就可以调用这个宏，例如:

```
%LET thedat=sasuser.class;
%LET varlist=name height weight;
%topr
```

注意调用宏时仍使用“%*宏名*”的格式，不像宏变量引用那样要用“&”来引用。调用宏时也不需要分号，因为其定义中已经有表示语句结束的分号。

我们从上面的例子发现这样定义的宏还是不太方便，需要定义了内容中需要的宏变量后才能调用宏。实际上，宏还有一种带参数(自变量)的定义格式，即:

%MACRO *宏名* (参数表);
...(定义宏的内容)
%MACRO *宏名*;

然后就可以按“%*宏名*(参数)”的形式调用。例如，上面的列表的程序利用带参数的宏可以改写为:

```
%MACRO topr(thedat, varlist);
      proc print data=&thedat. noobs label;
        var &varlist.;
      run;
%MEND topr;

%topr(sasuser.class, name height weight)
```

四. 流程控制结构

SAS 宏的一个重要功能是完成一些单调的工作的重复。重复工作的办法是利用宏DO循环，例如:

```
%MACRO mac;
%DO i=1 %TO 10;
  %PUT &i.;
%END;
%MEND mac;

%mac
```

在执行循环时产生了宏变量i并且依次赋值1,2,...,10。宏DO循环只能用于宏的定义内部。

本小节开始所举的实例就是利用了宏DO 循环。第一个循环定义了循环宏变量NP，并在循环内利用NP生成了保存输入文件名的变量ff和保存结果数据集名的变量fd。然后的data语句中只要适当替换即可。第二个循环产生了后面的纵向合并用的SET语句。注意宏赋值语句后面的值如果需要连接几个字符串只要直接连写就可以，不需要使用连接字符串用的运算符。

宏定义中也可以有分支结构，格式为"%IF *条件* %THEN *宏语句*" 或"%IF *条件* %THEN *宏语句* %ELSE *宏语句*"。其中的条件一般是判断相等(用等于号)。宏语句可以是复合的, 格式为:

%DO;
    ...(多个语句)
%END;

五. 宏与数据步的信息交换

SAS的工作模式一般不利于进行渐进式的分析，即先得到一个中间结果，再把中间结果作为下一步的

输入来进行分析。后面要讲的S语言最有利于这种工作模式。SAS的宏功能可以部分弥补SAS的这一缺陷，我们可以用宏变量在不同的数据步、过程步之间传递信息。

数据步的SYMPUT 函数是一个特殊的子程序：它能利用数据步运行中的变量值给一个宏变量赋值。SYMGET 也用于数据步中，作用和SYMPUT相反，可以在数据步执行时得到宏变量的值。注意这两个子程序的功能和宏变量赋值及宏变量引用不同，宏变量赋值、引用是静态的，是在数据步运行前就已经完成的，而SYMPUT 和SYMGET 可以在数据步运行中改变或访问宏变量的值。包含SYMPUT和SYMGET 的程序需要包在宏的定义中，否则对宏变量的引用是静态引用。

所谓子程序是类似于函数的结构，但是子程序不返回函数值，而且必须用CALL语句调用，见下面的例子。

下面举例说明。假设我们有一个数n放到了数据集A中，我们希望在程序中用到这个数，比如，把这个数用在脚注(footnote) 中，可以用如下程序：

```
data a;
  input n;
  cards;
10
;
run;

%MACRO minter;
  data _null_;
    set a;
    call symput('thenum', n);
```

```
  run;
  footnote "共&thenum.个结果";
%MEND minter;
%minter
proc print;run;
```

这里为了容易理解只把从数据集中读出的数用在了一个FOOTNOTE 语句中。在实际中，我们可以在数据步中计算出一个值然后用SYMPUT 把它保存入宏变量中。产生的宏变量可以用来控制循环次数，数据步中的变量个数，等等。见下例:

```
%MACRO minter2;
data b;
  nn = 1 + floor(10*uniform(0));
  call symput('nn', nn);
run;

data c;
  do i=1 to &nn;
    output;
  end;
run;
proc print;run;
%MEND minter2;
%minter2
```

SYMGET子程序用到的地方不多，这里不介绍了。

六. 宏语句和宏函数

这里简单提一下SAS宏语言所用的一些语句和函数，需要时读者再参考更详细的资料。

宏语句表:

%* *注释*	提供对宏的注释。它与普通注释的区别是普通注释会生成注释文本而宏注释在替换时不产生文本。
%DISPLAY	显示宏窗口。可以很容易地开发出窗口界面的用户交互程序。
%DO ...%END	复合语句。
%DO *循环变量=起始值* %TO *终止值*	计数循环
%DO %WHILE	当型循环
%DO %UNTIL	直到型循环
%GLOBAL	创建全程宏变量
%GOTO	宏程序跳转到标号位置
%IF ...%THEN ...%ELSE ...	分支结构
%INPUT	给宏变量输入值
%KEYDEF	定义显示管理功能键
%*标号*:	标记一个位置供%GOTO转移时识别位置
%LET	为宏变量赋值
%LOCAL	创建局部宏变量
宏调用	用“%*宏名*”或“%*宏名*(*参数*)”的格式调用已定义的宏。
%MACRO ...%MEND	定义宏。
%PUT	显示到记录窗口
%SYSEXEC	执行主机系统命令
%WINDOW	定义宏窗口

在宏函数中,%EVAL 可以计算算术和逻辑表达式。因为宏操作主要是进行字符串替换，而且字符串值中可以包含%和&这样的引用运算, 所以SAS宏提供了多个处理字符串运算和引用的函数。这些函数比较复杂，我们这里只罗列一下，需要的读者可以阅读专门的资料。

SAS宏语言中关于字符串的函数包括：%INDEX, %LENGTH, %QSCAN, %QSUBSTR, %QUPCASE, %SCAN, %SUBSTR, %UPCASE, %BQUOTE, %NRBQUOTE, %NRQUOTE, %NRSTR, %QUOTE, %STR, %SUPERQ, %UNQUOTE。

2.3.11 用PROC SQL管理数据

SAS系统首先是一个数据管理系统, 因此它除了可以用SAS语言程序管理SAS数据库、数据集外, 还提供了其它大型数据库管理系统(如Oracle、 Sybase)通用的SQL语言功能, 在SAS系统中SQL语言实现在SQL过程中。SAS 的 SQL 过程可以从一个或多个表中查询信息, 生成表, 向表中插入行, 更新表的内容, 对表进行纵向合并、横向连接等等。另外, PROC SQL还可以直接连接外部数据库, 我们已经在2.3.4作了介绍。SQL语言可以实现极其复杂的数据管理功能, 在这里我们只对它的查询功能作简单介绍, 感兴趣的读者可以自己阅读一些数据库管理方面的书籍。

用PROC SQL作查询的最简单的用法如下:

PROC SQL;
SELECT *第一项, 第二项, ..., 第n项*
FROM *数据集*
WHERE *观测选择条件*;
RUN;

其中SELECT是一个语句, FROM和WHERE叫做子句, 注意语句是在最后结尾的, 中间没有分号。SELECT子句中指定的各项一般为变量名, 中间用逗号分隔(注意不是用空格分隔)。FROM子句指定要从哪个数据集查询。WHERE子句指定选择观测的条件。所以, SELECT语句可以很方便地从一个表查询一个子集, 并可以自动输出到输出窗口而不需再使用PROC PRINT。例如, 下面的程序显示语文成绩在100分以上(包含)的学生的姓名和数学成绩:

```
proc sql;
  select name, math
    from c9501
    where chinese>=100;
run;
```

结果显示

NAME	MATH
张红艺	89
张聪	98
刘颖	80

在SELECT语句中还可以加入ORDER BY子句, 可以为查询结果排序。比如, 下程序

```
proc sql;
  select name, math
    from c9501
    where chinese>=100
    order by math desc;
run;
```

结果为

NAME	MATH
张聪	98
张红艺	89
刘颖	80

SELECT的强大查询功能还表现在它可以从几个表联合查询。比如, 考虑2.3.8中的C9501X和C9501Y, 我们要从这两个数据集查询与从C9501一个数据集同样的结果, 可以用此程序:

```
proc sql;
  select c9501x.name, math
    from c9501x, c9501y
    where c9501x.name=c9501y.name
          and chinese>=100
    order by math desc;
run;
```

其中连接两个数据集的办法是在WHERE子句指定C9501X. NAME = C9501Y. NAME这样的连接条件。在SELECT中指定变量时如果有两个数据集中共有的变量要用C9501X.NAME这样的带有表名(数据集名)的形式。

连接的两个表有时是同一个表。比如, 我们有几个学生的姓名和生日, 希望找出那些有相同生日的人。可以用如下的SQL过程:

```
title '找出生日相同的人';
data class;
  input name $ 1-8 birth yymmdd10.;
  format birth yymmdd10.;
  label name='姓名'  birth='生日';
  cards;
李明       78-6-1
王思明     78-5-19
张聪       78-6-1
刘颖       78-10-18
张红艺     79-5-19
;
proc sql;
  select name, birth
    from class a
    where birth in select birth
      from class b
      where b.name ^= a.name
    order by a.birth;
run;
```

结果如下:

```
     找出生日相同的人

   姓名              生日
   --------------------
   李明       1978-06-01
   张聪       1978-06-01
```

如果我们还希望把查询的结果存入一个数据集, 可以在上面的第一个SELECT语句前面加上“CREATE TABLE *表名* AS”:

```
proc sql;
  CREATE TABLE bsame AS
    select name, birth
      from class a
      where birth in select birth
        from class b
        where b.name ^= a.name
      order by a.birth;
run;
proc print data=bsame label;
  id name;
  by birth;
run;
```

结果如下:

```
------ 生日=1978-06-01 --------

         姓名

         李明
         张聪
```

上面求生日要求年月日完全相同, 如果我们只要求月日相同可以使用三个SQL语句: CREATE TABLE语句生成一个表示月日的变量MD存入临时数据集DATAMD, SELECT语句列出同生日人名单, DROP TABLE语句删除临时数据集DATAMD。

```
proc sql;
  create table datamd as
    select name, birth, month(birth)*100+day(birth) AS md
      from class;
  select name, birth
    from datamd a
    where md in select md
```

```
      from datamd b
      where b.name ^= a.name
    order by a.md;
  drop table datamd;
run;
```

结果为:

```
     找出生日相同的人

   姓名            生日
   --------------------
   王思明     1978-05-19
   张红艺     1979-05-19
   李明       1978-06-01
   张聪       1978-06-01
```

如果不用SQL过程想得到同样的结果, 可以使用如下数据步和过程步:

```
proc freq data=class noprint;
  tables birth / out=bfreq;
run;
proc sort data=class;
  by birth;
proc sort data=bfreq;
  by birth;
data bsame;
  merge class bfreq;
  by birth;
  if count>1;
run;
proc print data=bsame label noobs;
  var name;
  by birth;
run;
```

练习

1. 用SAS数据步列出10000以下的素数, 写出程序。

2. 生成t分布的双侧分位数表。水平取0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.10, 0.20, 自由度取1–100, 分位数精确到小数点后3位。表格应为行、列对齐的形式, 并有列标题。写出生成这样的表格并存放到一个文本文件中的SAS程序。

3. 写出计算从自己生日到2000年初经过的天数的程序。

4. 下表为某邮购服务部的部分顾客记录:

姓名	性别	地区	日期	金额
章文	男	华东	1996-3-20	1099
王国铭	男	华东	1996-5-19	39
童子敏	女	华北	1996-1-5	986
刘念新	男	东北	1997-10-1	3581
李思今	女	华北	1997-4-4	659
关昭	女	东北	1996-11-5	358
赵霞	女	东北	1998-9-6	2010

(1)用数据步把此数据输入到SAS数据集;

(2)用程序找出男性顾客购买金额超过1000的哪些人;

(3)把数据拆分为包含姓名、性别、地区的一个数据集和包含姓名、日期、金额的一个数据集;

(4)用MERGE和BY合并上一步拆开的两个数据集。

第三章 SAS功能基础

前面讲过, SAS系统用SAS数据步生成和管理数据, 用过程步进行分析、报表、绘图。本章先介绍SAS过程步的一般用法和常用语句的含义, 然后讲解SAS的简单报表、分析、绘图功能的使用，最后介绍SAS的分析员(Analyst)模块。

3.1 SAS过程初步

3.1.1 SAS过程用法

SAS过程步的一般形式为：

PROC *过程名* DATA=*输入数据集* *选项*;
过程语句 / 选项;
过程语句 / 选项;
.....
RUN;

其中PROC语句的选项是可选的, 用来规定过程运行的一些设置, 如果有多个选项用空格分开。DATA = 输入数据集也是可选的, 如果缺省的话使用最近生成的数据集。过程步一般以RUN语句结束, 也可以省略RUN语句而在下一个过程步或数

据步的开始处结束, 另外还有一种所谓“交互式过程”可以在遇到RUN语句时不结束过程运行, 只有遇到QUIT 语句或者下一个过程步、数据步时才结束。过程步在PROC 语句之后、结束之前可以有若干个过程语句。通常情况下, 过程语句与数据步中的语句不同, 数据步中的语句不能用在过程步中。过程步语句一般以某一个关键字开头, 比如VAR、BY、TABLES、WEIGHT等, 语句中有一些有关说明, 如果有选择项的话要写在斜杠后。

SAS过程步有些是对数据集作某种变换(比如SORT过程对数据集排序), 不生成显示结果；多数过程步是对数据集作某些分析、报表, 这时结果出现在OUTPUT窗口(高精度绘图过程的输出在GRAPHICS窗口)。对OUTPUT窗口的结果, 我们可以用“File - Save As” 菜单把它保存到一个文本文件进行进一步的修饰, 插入到其它报告中, 也可以用“File - Print”菜单之间打印。

3.1.2 SAS过程步常用语句

本小节简单介绍几个在SAS过程步中常见的语句, 更具体的用法可以在以后实际用到时再仔细体会。

一、VAR语句

VAR语句在很多过程中用来指定分析变量。在VAR后面给出变量列表：

VAR *变量名1 变量名2 ... 变量名n*;

变量名列表可以使用省略的形式, 如X1−X3,

MATH−−CHINESE等。VAR 用法例如：

```
var math chinese;
```

二、MODEL语句

MODEL语句在一些统计建模过程中用来指定模型的形式。其一般形式为

MODEL *因变量* = *自变量表* / *选项*;

比如

```
model math=chinese;
```

即用语文成绩预测数学成绩。

三、BY语句和CLASS语句

BY语句在过程中一般用来指定一个或几个分组变量, 根据这些分组变量值把观测分组, 然后对每一组观测分别进行本过程指定的分析。在使用带有BY语句的过程步之前一般先用SORT过程对数据集排序。比如, 假设我们已经把C9501数据集按性别排序, 则下列PRINT过程可以把男、女生分别列出：

```
proc print data=c9501;
  by sex;
run;
```

结果为

```
                        The SAS System                    2

      ---------------------- SEX=男 ----------------------
```

```
                OBS     NAME     MATH    CHINESE

                 1      李明      92       98
                 2      王思明    86       90
                 3      张聪      98       109


---------------------- SEX=女 ----------------------

                OBS     NAME     MATH    CHINESE

                 4      张红艺    89       106
                 5      刘颖      80       110
```

在一些过程(如方差分析)中, 使用CLASS语句指定一个或几个分类变量。而在另一些过程(如MEANS)中, CLASS语句作用与BY语句类似, 可以指定分类变量, 把观测按分类变量分类后分别进行分析。使用CLASS时不需要先按分类变量排序。

四、OUTPUT语句

在过程步中经常用OUTPUT 语句指定输出结果存放的数据集。不同过程中把输出结果存入数据集的方法各有不同, OUTPUT语句是用得最多的一种, 其一般格式为:

OUTPUT OUT=输出数据集名 关键字=变量名 关键字=变量名...;

其中用OUT=给出了要生成的结果数据集的名字, 用"关键字=变量名"的方式指定了输出哪些结果(关键字的例子比如MEANS过程中的MEAN, VAR, STD等等), 等号后面的变量名指定了这些结果在输出数据集中叫什么名字。例如

```
proc means data=sasuser.c9501;
  var math;
```

```
  output out=result n=n
         mean=meanmath var=varmath;
run;
proc print data=result; run;
```

五、FREQ语句和WEIGHT语句

FREQ语句指定一个重复数变量, 每个观测中此变量的值说明这个观测实际代表多少个完全相同的重复观测。FREQ 变量只取整数值。如

```
  freq  numcell;
```

WEIGHT语句指定一个权重变量, 在某些允许加权的过程中代表权重, 其值与观测对应的方差的倒数成比例。

六、ID语句

有些过程(如PRINT、UNIVARIATE)需要输出观测的代号, 这一般使用观测的序号。但是, 如果数据集中有一个变量可以用来区分观测(如人名、省市名), 就可以用ID语句指定这个变量作为观测标识, 如：

```
  id name;
```

指定用变量NAME的值来标识观测。

七、WHERE语句

用WHERE语句可以选择输入数据集的一个行子集来进行分析, 在WHERE 关键字后指定一个条件。比如：

```
  where math>=60 and chinese>=60;
```

指定只分析数学、语文成绩都及格的学生。

八、LABEL语句和FORMAT语句

过程步中的LABEL语句为变量指定一个临时标签, 很多过程可以使用这样的标签。LABEL语句的格式为

LABEL 变量名＝'标签' 变量名＝'标签' ...;

例如

```
proc print data=sasuser.c9501 label;
  id name;
  var math chinese;
  label name='姓名' math='数学成绩'
        chinese='语文成绩';
run;
```

结果显示

姓名	数学成绩	语文成绩
李明	92	98
张红艺	89	106
王思明	86	90
张聪	98	109
刘颍	80	110

FORMAT语句可以为变量输出规定一个输出格式, 比如

```
proc print data=sasuser.c9501;
  format math 5.1 chinese 5.1;
run;
```

使得列出的数学、语文成绩宽度占5位, 带一位小数。

事实上, 在生成数据集的DATA步中也可以用FORMAT语句规定变量的输出格式, 用LABEL语句规定变量的标签, 用LENGTH语句规定变量的存贮长度, 用ATTRIB语句同时规定变量的各属性。在数据步中规定的变量属性是附属于数据集本身的, 是永久的；在过程步中规定的变量属性(标签、输出格式等)只用于此过程的本次运行。

3.2 列表报告

PRINT 过程用来生成列表报告。本节讲解PRINT过程的使用, 并用它来辅助讲解一些常用语句的使用及SAS输出管理。

3.2.1 基本用法

PRINT过程是最常用的SAS过程之一。我们在生成了一个数据集之后, 如果不是太大, 一般都用一个

```
proc print;run;
```

过程步来列出数据集的内容, 这样可以检查变量与值之间对应是否正确, 数据输入是否正确。为了列出一个指定的数据集, 在PROC 语句中使用DATA＝选项指定要列表的输入数据集名, 比如：

```
proc print data=sasuser.gpa; run;
```

在过程内使用VAR语句可以指定要列出的变量并指定顺序。比如,

```
proc print data=c9501;
  var name chinese sex;
run;
```

列出变量NAME、CHINESE、SEX的值。注意这已不是生成时的变量顺序。变量MATH未列出。结果如下

```
                The SAS System                          3
    OBS     NAME      CHINESE     SEX

     1      李明          98       男
     2      张红艺       106       女
     3      王思明        90       男
     4      张聪         109       男
     5      刘颖         110       女
```

注意PRINT的输出第一列总是标为OBS, 值为观测序号。我们有时不想输出这一列, 可以在PROC PRINT语句中加入NOOBS选项, 如：

```
proc print data=c9501 noobs;
run;
```

结果中就没有了OBS这一列。

在过程中使用WHERE 语句可以从输入数据集中选一个子集来处理, 在PRINT 过程中使用WHERE 可以指定只列出满足条件的观测。比如,

```
proc print data=c9501;
  where name in ('李明', '张聪');
run;
```

结果为

OBS	NAME	SEX	MATH	CHINESE	AVG
1	李明	男	92	98	86.8333
4	张聪	男	98	109	94.4167

只列出了李明和张聪两个人的观测。注意其观测序号分别为1和4, 这是生成C9501 数据集时确定的。

3.2.2 SAS中对输出结果的管理

SAS的输出都显示在输出窗口。在运行了多个过程后，输出窗口积累了多个过程的输出，有时不易找到或特定的结果。新版本的SAS 系统提供了一个结果管理窗口来管理输出，叫Results 窗口。这个窗口缺省是打开的，固定放置在运行环境的左半部分，如果没有可以从“View - Results” 菜单打开。

在运行了如下程序后，结果管理窗口的样子见图3.1。

```
proc print data=c9501;
run;
```

```
proc means data=c9501;
  var math chinese;
run;
```

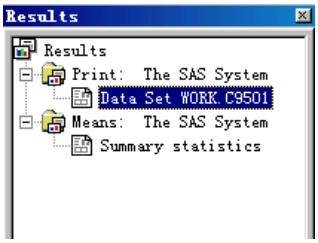


图 3.1: 结果管理窗口

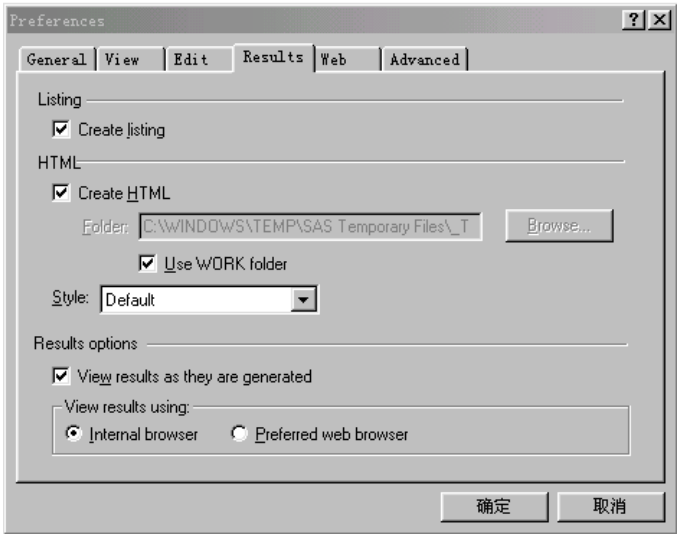


图 3.2: 打开HTML输出

过程的输出以树形目录方式显示，双击某一个结果项可以直接跳到该结果。还可以在这个窗口中删除、保存某些结果，选定某个结果用右键弹出菜单可以选择此窗口的管理功能。

SAS除了一般文本输出外还提供了HTML格式(即网页格式)的输出。要得到这样的输出，需要调用“Tools - Options - Preferences”菜单，在弹出的对话框中找到“Results”页，选中其中的“Create HTML”，见图3.2。

再次运行上面的程序，这时每个过程都生成了两种输出结果：一般文本输出和HTML输出。结果管理窗口见图3.3。单击其中的加号打开折叠

的输出项，双击其中PRINT 过程的HTML输出，显示出图3.4的HTML结果。对这样的输出Results窗中甚至还提供了一个Edit Source 弹出菜单项可以把HTML结果的源代码打开到程序编辑窗口进行编辑。

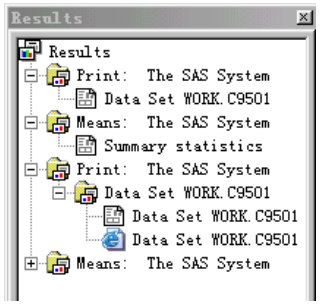


图 3.3: 结果管理窗口

The SAS System

Obs	name	sex	math	chinese	avg
1	李明	男	92	98	86.8333
2	刘颖	女	80	110	85.8333
3	王思明	男	86	90	80.5000
4	张聪	男	98	109	94.4167
5	张红艺	女	89	106	88.6667

图 3.4: HTML输出示例

3.2.3 使用中文列标题

为了对列标题使用中文, 可以在过程内用LABEL 语句给变量指定标签, 同时在PROC PRINT 语句中加LABEL 选项。如果已经在生成数据集的数据步中用LABEL 语句或ATTRIB 语句为变量指定了标签则不必重复定义标签。例如：

```
proc print data=c9501 noobs label;
  var name sex math chinese avg;
  label name='姓名' sex='性别' math='数学'
        chinese='语文' avg='平均分';
run;
```

则结果如下：

```
            The SAS System                    6
    姓名        性别      数学      语文      平均分
    李明         男        92        98     86.8333
    张红艺       女        89       106     88.6667
    王思明       男        86        90     80.5000
    张聪         男        98       109     94.4167
    刘颖         女        80       110     85.8333
```

3.2.4 标题及全程语句

我们从上面的输出结果看到, 在每页输出结果上面有一行标题, 内容为“The SAS System”。事实上, 我们可以指定自己的标题来取代SAS缺省的标题。指定标题的TITLE语句的格式为：

TITLE '*标题内容*';

例如, 在前一例的程序前面加上一行

```
title '95级1班成绩表';
```

则输出结果的标题为“95级1班成绩表”。要注意的是, TITLE语句是一个所谓的“全程”语句, “全程”语句与一般语句不同, 一般语句必须用在数据步或过程步内, 作为数据步或过程步的一部分, 而全程语句则既可以用在数据步和过程步内, 又可以单独使用(在数据步、过程步外部)使用。全程语句的作用一般有持续性, 即：全程语句的效果将持续到退出SAS系统或用另一个同样的全程语句来修改它。比如, 我们在上面用TITLE语句指定了一个标题, 那么, 这个标题的作用将持续下去, 虽然下一个过程没有用TITLE语句指定标题它也会出现在下一个过程的输出中, 例如在

上面用TITLE语句为C9501数据集的列表输出加了标题后, 再运行如下程序：

```
proc means data=sasuser.gpa;
run;
```

你会发现标题“95级1班成绩表”仍出现在输出的每一页, 而这个标题已经与输出内容矛盾了(现在分析的是SASUSER. GPA数据集而不是C9501 数据集)。为了取消这个标题, 只要用一个没有规定内容的空TITLE语句, 即：

```
title;
```

这时连缺省的“The SAS System”标题也没有了。

用全程语句FOOTNOTE可以为输出加脚注, 如：

```
footnote '第三章例子输出';
```

则其后的输出每页下方会有脚注“第三章例子输出”, 直到用另一个FOOTNOTE 语句指定新的脚注, 或用空FOOTNOTE 语句取消脚注为止。

另一个全程语句OPTIONS语句可以规定系统运行的一些选择项, 比如输出是否每页有页号, 是否有日期, 输出的行宽, 输出每一页的高度(行数), 等等。其使用例如：

```
options nonumber nodate
        linesize=64 pagesize=60;
```

其中NONUMBER 表示输出不显示页号(改用NUMBER 则规定显示页号), NODATE 表示不在每页显示运行日期和时间(改用DATE 则显示), LINESIZE = 64规定输出每行最宽不超过64个字符(这是允许的最小行宽), PAGESIZE=60规定输出每页为60行, 不足时用空行补齐。

3.2.5 用BY语句分组处理

前面我们已经讲过, 在过程步中使用BY语句可以指定分类变量, 把观测分类处理。在使用带有BY语句的过程之前一般用SORT过程对数据集按照分类变量排序。例如:

```
proc sort data=c9501;
  by sex;
run;
proc print data=c9501;
  by sex;
run;
```

结果为:

```
-------------------------- SEX=男 -------------------------
        OBS     NAME     MATH    CHINESE      AVG

         1      李明       92        98      86.8333
         2      王思明     86        90      80.5000
         3      张聪       98       109      94.4167

-------------------------- SEX=女 -------------------------

        OBS     NAME     MATH    CHINESE      AVG

         4      张红艺     89       106      88.6667
         5      刘颖       80       110      85.8333
```

3.2.6 计算总计和小计

在PRINT 过程中可以用SUM 语句计算某个变量的总计(总和)。例如, 9501 班的同学购买课外书所用的钱数用如下程序输入数据集：

```
data bkmoney;
  input name $  amount;
  cards;
李明  20
张红艺 15
王思明 10
张聪 20
刘颍 50
;
run;
```

为了列出此数据集并计算全班的总书款, 只要在PRINT 过程中加上SUM 语句, SUM 语句中指定要求和的变量名AMOUNT：

```
proc print data=bkmoney noobs;
  sum amount;
run;
```

结果为：

```
 NAME      AMOUNT

李明          20
张红艺         15
王思明         10
张聪          20
刘颍          50
             ======
              115
```

可见总额为115元。

SUM语句中也可以指定多个变量名, 可以同时对这些变量求和。

用BY语句与SUM语句就可以既计算总和也计算分组小计。比如, 我们除了要计算学生购买课外书总支出外还想分男、女生计算总支出, 可以用下面的程序。注意由于数据集BKMONEY 中没有性别的信息, 我们用了带MERGE 语句的数据步来横向合并C9501和BKMONEY 两个数据集。

```
proc sort data=c9501;
  by name;
proc sort data=bkmoney;
  by name;
data c9501bk;
  merge c9501 bkmoney;
  by name;
run;
proc sort data=c9501bk;
  by sex;
proc print data=c9501bk;
  by sex;
  sum amount;
run;
```

程序的结果为：

```
-------------------------- SEX=男 -------------------------
   OBS     NAME     MATH    CHINESE      AVG      AMOUNT
     1     李明       92        98      86.8333      20
     2     王思明     86        90      80.5000      10
     3     张聪       98       109      94.4167      20
                                                    ------
   SEX                                               50


-------------------------- SEX=女 -------------------------
```

OBS	NAME	MATH	CHINESE	AVG	AMOUNT
4	刘颖	80	110	85.8333	50
5	张红艺	89	106	88.6667	15

SEX					65
					======
					115

可见总额为115, 男生小计为50, 女生小计为65。

3.3 汇总表格

PRINT过程可以制作列表, 它列出所有观测。当观测个数很多时, 这样的列表意义不大。TABULATE 过程制表不是列出观测, 而是计算观测的分类统计量, 绘制统计量的表格。这对于数据的汇总比较有用。

TABULATE可以作出很复杂的表, 其一般格式为:

```
PROC TABULATE DATA=数据集名;
    CLASS 分类变量;
    VAR 分析变量;
    TABLE 页维说明, 行维说明, 列维说明/ 选项;
RUN;
```

其中CLASS 语句给出分类变量, 用分类变量可以给观测分类, 计算统计量时可以对每一类分别计算。VAR语句给出区间变量。

TABLE语句规定了绘制什么样的表格。我们用例子说明。比如, 对C9501BK 数据集, 我们希望表中绘出男、女生的课外书支出总和, 可以用如下例子:

```
proc tabulate data=c9501bk;
  class sex;
  var amount;
  table sex, amount;
run;
```

结果为：

```
-------------------------------------
|                     |   AMOUNT    |
|                     |-------------|
|                     |    SUM      |
|---------------------+-------------|
|SEX                  |             |
|---------------------|             |
|男                   |       50.00|
|---------------------+-------------|
|女                   |       65.00|
-------------------------------------
```

由上例可见在TABLE语句中指定一个分类变量可以把表格按此变量的值分格, 指定一个区间变量可以计算其和。因为变量SEX 和AMOUNT 中间用逗号分隔, 所以SEX 在行维, 表格的行用SEX 的值区分, AMOUNT 在列维, 它画在列标题中。如果只是想统计男女生人数，可以只用SEX 一个变量，如:

```
proc tabulate data=c9501bk;
  class sex;
  table sex;
run;
```

区间变量的缺省统计量是总和，分类变量的缺省统计量是频数。如果我们要计算其它统计量, 可以用“变量名*统计量名”的形式。统计量名包括N, NMISS, MEAN, STD, MIN, MAX, RANGE, SUM, USS,

CSS, STDERR, CV, T(检验均值为0的t统计量值), PRT(t统计量的p值), VAR, SUMWGT(权数变量的和), PCTN(某类观测占总观测个数的百分比), PCTSUM(某类观测的总和占全部总和的百分比)。例如, 我们可以用如下程序求男、女生的数学、语文成绩平均值及标准差：

```
proc tabulate data= c9501bk;
  class sex;
  var math chinese;
  table sex, (math chinese)*(mean std);
run;
```

结果为：

```
-------------------------------------------------------------------
|                |         MATH           |         CHINESE        |
|                |------------------------+------------------------|
|                |    MEAN    |    STD    |    MEAN    |    STD    |
|----------------+------------+-----------+------------+-----------|
|SEX             |            |           |            |           |
|----------------|            |           |            |           |
|男              |      92.00|       6.00|       99.00|       9.54|
|----------------+------------+-----------+------------+-----------|
|女              |      84.50|       6.36|      108.00|       2.83|
-------------------------------------------------------------------
```

我们看到, 要并列变量或并列统计量的话只要把各项用空格连接, 如“MATH CHINESE”和“MEAN STD”。变量和统计量名之间用星号连接。变量名和统计量名的次序也可以颠倒过来，比如(mean std)*(math chinese)，得到的表格中也将以统计量为大栏目而以变量为小栏目。

上面的表格只分类计算了统计量值, 如果要计算总的统计量值, 只要加一个ALL关键字。把上面的例子中的TABLE语句改成：

```
table sex all, (math chinese)*(mean std);
```

结果为

```
-------------------------------------------------------------------
|                |          MATH          |         CHINESE        |
|                |------------------------+------------------------|
|                |    MEAN    |    STD    |    MEAN    |    STD    |
|----------------+------------+-----------+------------+-----------|
|SEX             |            |           |            |           |
|----------------|            |           |            |           |
|男              |       92.00|       6.00|       99.00|       9.54|
|----------------+------------+-----------+------------+-----------|
|女              |       84.50|       6.36|      108.00|       2.83|
|----------------+------------+-----------+------------+-----------|
|ALL             |       89.00|       6.71|      102.60|       8.47|
-------------------------------------------------------------------
```

事实上, 我们用星号连接分类变量和分类变量可以构成交叉分组, 用星号连接分类变量和区间变量可以分类计算区间变量的统计量。例如, 要考察性别的频数分布, 使用统计量N和PCTN, TABLE语句可以写成

```
table (sex all)*(N PCTN);
```

结果如下：

```
-------------------------------------------------------------------------------
|                      SEX                      |                       |
|-----------------------------------------------|                       |
|          男          |          女            |          ALL          |
|----------------------+------------------------+-----------------------|
|     N      |   PCTN  |     N      |   PCTN    |     N      |   PCTN   |
|------------+---------+------------+-----------+------------+----------|
|        3.00|    60.00|        2.00|      40.00|        5.00|    100.00|
-------------------------------------------------------------------------------
```

如果不是对分类变量本身计算N、PCTN这样的统计量而是分类计算其他变量的统计量, 则还需要指定需要统计的区间变量, 例如TABLE语句写成

```
  table (sex all)*math*(mean std);
```

则结果为

```
-------------------------------------------------------------------------
|                      SEX                          |                   |
|---------------------------------------------------|                   |
|          男          |          女          |          ALL          |
|----------------------+----------------------+-----------------------|
|         MATH         |         MATH         |         MATH          |
|----------+-----------+----------+-----------+----------+------------|
|   MEAN   |    STD    |   MEAN   |    STD    |   MEAN   |    STD     |
|----------+-----------+----------+-----------+----------+------------|
|      92.00|      6.00|      84.50|      6.36|      89.00|      6.71|
-------------------------------------------------------------------------
```

以上两个例子没有行维, 只有列维。

可以在TABULATE 过程中使用KEYLABEL 语句指定各统计量的标签。其格式为

KEYLABEL 关键字＝'标签';

例如, 下例用KEYLABEL 指定了MEAN 和STD 的标签, 又用LABEL 语句指定了变量的标签：

```
proc tabulate data=c9501bk;
  class sex;
  var math chinese;
  table (sex all),
        (math chinese)*(mean std);
  keylabel mean='平均值'
           std='标准差' all='总计';
  label sex='性别' math='数学'
        chinese='语文';
run;
```

另一种指定统计量标签的办法是在TABLE 语句中直接指定，例如上例可以写成：

```
proc tabulate data=c9501bk;
  class sex;
  var math chinese;
  table (sex 'all'='总计'),
        (math chinese)*('mean'='平均值'
         'std'='标准差');
  label sex='性别' math='数学'
        chinese='语文';
run;
```

结果为

```
------------------------------------------------------------------------
|                    |          数学          |          语文          |
|                    |------------------------+------------------------|
|                    |  平均值   |  标准差    |  平均值   |  标准差    |
|--------------------+-----------+------------+-----------+------------|
|性别                |           |            |           |            |
|--------------------|           |            |           |            |
|男                  |      92.00|        6.00|      99.00|        9.54|
|--------------------+-----------+------------+-----------+------------|
|女                  |      84.50|        6.36|     108.00|        2.83|
|--------------------+-----------+------------+-----------+------------|
|总计                |      89.00|        6.71|     102.60|        8.47|
------------------------------------------------------------------------
```

我们得到了一个全中文的统计表格。

注意：有时用TABULATE过程作表时表格线是一些乱码, 这是SAS系统设置文件CONFIG.SAS中关于FORMCHAR的设置的问题。在SAS系统的安装目录下查找到一个名为SASV8.CFG 的文件(第8版以前的SAS 系统用CONFIG.SAS), 把其中的所有以“−FORMCHAR”开始的行都注释掉, 只留下如下的行不注释：

```
-FORMCHAR "|----|+|---+=|-/\<>*"
```

就可以解决这个问题。

3.4 数据排序

在SAS过程中用BY语句可以把观测分类进行处理, 但在此之前需要先用SORT过程排序。SORT过程可以把数据集按某一个或若干个变量的次序进行排序。比如, 我们要把数据集C9501的观测(行)按性别排序(分类), 可以用此程序：

```
proc sort data=c9501;
  by sex;
run;
```

注意这样用DATA=指定的数据集既是输入数据集又是输出数据集。过程的结果在输出窗口没有显示, 只是把数据集按要求进行了排序。

可以按几个变量排序, 比如, 要按男、女性别排序, 并在男生、女生内部按平均分由高到低排序, 可以这样作：

```
proc sort data=c9501;
  by sex descending avg;
run;
```

BY语句内在一个变量名前面加上DESCENDING 关键字表示此变量的排序是由大到小的。

排序结果可以用“proc print;run;”显示出来。

在INSIGHT 中我们也可以对数据集排序, 请自己复习如何在INSIGHT 中对数据集排序。

3.5 数据集转置

一个SAS数据集可以看成是某种矩阵，观测为矩阵

的行，变量的列。有时我们需要改变观测与列的关系，这可以用TRANSPOSE过程来实现。

先举一个合并观测的例子。假设我们对若干个病人先后试验了A药和B药，药效记录在val变量中，数据见表3.1(为讲解方便我们虚构了数据)。为了进行统计分析我们需要把两次的药效存入两个变量, 即两个观测合并为一个观测, 用如下程序可以实现：

```
proc sort data=onecol;
  by num;
run;
proc transpose data=onecol out=twotest;
  var val;
  id test;
  by num;
run;
```

转置后的结果数据集见表3.2。输出数据集把原来每个人(用NUM标识)的两个VAL变量值变成了一个观测的两个变量, VAR语句指定了要转置的变量, ID语句指定观测变成变量时的变量名, BY语句规定在分组以后对组内的行、列进行转置。输出数据集新增的变量_NAME_指出某一观测对应于原数据集的哪一列。以上例子的方法是稳健的, 即使打乱了原始数据集的次序或删除某些观测最后的结果仍是正确的。

再来考虑合并的逆操作, 即拆分观测。假设我们对每个人有两个测量值TEST1和TEST2, 见表3.3, 希望把这两个测量合并到同一列, 增加一列来区分观测来自哪一个变量, 可以用如下程序:

```
proc sort data=twocol;
  by num;
run;
```

```
proc transpose data=twocol out=onetest;
  var test1 test2;
  by num;
run;
```

其中VAR语句指定要转置的两列, BY语句要求在组内进行转置。结果见表3.4, 新增的变量_NAME_用来区分每一新生成的观测对应于原数据集的何变量, 拆分后的变量TEST1和TEST2统一命名为COL1, 这是系统缺省给出的行转置变成列后的列名。

上面的两个例子是分组后要转置的矩阵只有一列或只有一行的特殊情况。转置当然也可以是同时对多行和多列进行, 比如下面的例子中原始数据集为4行3列, 转置后变成了3行4列。程序如下:

```
data mat;
  input x1 x2 x3;
  cards;
1   2  3
4   5  6
7   8  9
10 11 12
;
run;
```

表 3.1: 待合并观测的数据集ONECOL

病历号 NUM	药物 TEST	药效 VAL
1	a	11
2	a	12
3	a	13
1	b	21
2	b	22
3	b	23

表 3.2: 合并观测后的数据集TWOTEST

病历号 NUM	来源变量 _NAME_	药物A A	药物B B
1	val	11	21
2	val	12	22
3	val	13	23

```
proc transpose data=mat out=matt;
  var x1 x2 x3;
run;
proc print;run;
```

结果如下:

```
Obs    _NAME_    COL1    COL2    COL3    COL4

 1       x1        1       4       7       10
 2       x2        2       5       8       11
 3       x3        3       6       9       12
```

结果中的_NAME_变量保存了每一行在原始数据集中的列名(变量名), 而原始数据集中的四行则变成了结果中的COL1–COL4共四列。

3.6 描述统计

MEANS、UNIVARIATE和FREQ这三个过程用来计算简单的描述统计量。MEANS 和UNIVARIATE 过程对区间变量计算均值、标准差等数字特征, 而FREQ 过程对离散变量计算取值频数分布。

表 3.3: 待拆分观测的数据集TWOCOL

病历号 NUM	测量1 TEST1	测量2 TEST2
1	11	21
2	12	22
3	13	23

表 3.4: 拆分观测后的数据集ONETEST

病历号 NUM	来源变量 _NAME_	测量 COL1
1	test1	11
1	test2	21
2	test1	12
2	test2	22
3	test1	13
3	test2	23

例如, 我们要对C9501 中的数学成绩、语文成绩计算简单统计量, 只要用如下MEANS 过程：

```
proc means data=c9501;
  var math chinese;
run;
```

结果为

```
Variable  N          Mean     Std Dev     Minimum      Maximum
------------------------------------------------------------
MATH      5    89.0000000   6.7082039  80.0000000   98.0000000
CHINESE   5   102.6000000   8.4734881  90.0000000  110.0000000
------------------------------------------------------------
```

如果使用UNIVARIATE过程则可以计算较多的统计量, 例如我们分析SASUSER. GPA中的变量GPA, 可以用如下程序：

```
proc univariate data=sasuser.gpa;
  var gpa;
run;
```

结果为

```
                  The UNIVARIATE Procedure
                         Variable:  gpa

                            Moments

 N                          224    Sum Weights               224
 Mean                4.63522321    Sum Observations      1038.29
 Std Deviation       0.77939493    Variance            0.60745645
 Skewness            -0.6895022    Kurtosis            0.36481671
 Uncorrected SS       4948.1687    Corrected SS        135.462789
 Coeff Variation     16.8146148    Std Error Mean      0.05207551


                   Basic Statistical Measures
```

Location		Variability	
Mean	4.635223	Std Deviation	0.77939
Median	4.740000	Variance	0.60746
Mode	5.060000	Range	3.88000
		Interquartile Range	1.05000

Tests for Location: Mu0=0

Test	-Statistic-		-----p Value------	
Student's t	t	89.00965	Pr > |t|	<.0001
Sign	M	112	Pr >= |M|	<.0001
Signed Rank	S	12600	Pr >= |S|	<.0001

Quantiles (Definition 5)

Quantile	Estimate
100% Max	6.000
99%	6.000
95%	5.730
90%	5.610
75% Q3	5.215
50% Median	4.740
25% Q1	4.165
10%	3.660
5%	3.110
1%	2.400
0% Min	2.120

Extreme Observations

----Lowest----		----Highest---	
Value	Obs	Value	Obs
2.12	159	5.9	93
2.39	75	6.0	48
2.40	13	6.0	49
2.58	221	6.0	141
2.65	87	6.0	188

输出包括五个部分。第一部分是矩统计量，各统计量已在1.3.7中作了介绍。第二部分为基本的位置和分散程度统计量，位置统计量包括均值、中位数、众数，分散程度统计量包括标准差、方差、极差、四分位间距。第三部分为关于均值等于零的三种检验的结果，包括t检验、符号检验和符号秩检验。第四部分为各个重要的分为数估计。第五部分是变量的五个最低值和五个最高值。

FREQ过程可以考察离散变量的取值分布, 在TABLES 语句中指定要分析的变量。比如, 我们想了解C9501 中性别的分布情况, 可以用：

```
proc freq data=c9501;
  tables sex;
run;
```

结果为

SEX	Frequency	Percent	Cumulative Frequency	Cumulative Percent
男	3	60.0	3	60.0
女	2	40.0	5	100.0

可见FREQ的分析结果包括变量每个值的取值频数(观测个数)、百分比、累积频数、累积百分比。也可以对区间变量使用FREQ 过程列出频数分布, 如

```
  tables math;
```

MEANS、UNIVARIATE、FREQ的结果可以在INSIGHT 的“Analyze - Distribution” 和“Tables - Frequency Table”得到。

3.7 相关系数计算

CORR过程用来计算变量的相关系数。相关系数可以反映变量两两之间的线性相关。比如, 为了计算SASUSER. GPA 中的三个变量HSM, HSS, HSE 两两之间的相关系数(普通的Pearson 相关系数), 只要用如下程序：

```
proc corr data=sasuser.gpa;
  var hsm hss hse;
run;
```

结果如下：

```
                    The CORR Procedure

          3  Variables:    hsm      hss      hse


                    Simple Statistics

 Variable           N          Mean       Std Dev           Sum

 hsm              224       8.32143       1.63874          1864
 hss              224       8.08929       1.69966          1812
 hse              224       8.09375       1.50787          1813

                    Simple Statistics

             Variable       Minimum       Maximum

             hsm            2.00000      10.00000
             hss            3.00000      10.00000
             hse            3.00000      10.00000


         Pearson Correlation Coefficients, N = 224
                Prob > |r| under H0: Rho=0

                      hsm           hss           hse

        hsm       1.00000       0.57569       0.44689
```

```
                                  <.0001          <.0001

         hss          0.57569     1.00000         0.57937
                       <.0001                      <.0001

         hse          0.44689     0.57937         1.00000
                       <.0001      <.0001
```

输出分为两个部分, 第一部分是各变量的简单统计量, 第二部分是三个变量两两之间的相关系数矩阵。比如, HSM 和HSS 之间的相关系数为0.57569, 这个相关系数为零的显著性概率值为0.0001。表格共有三行三列, 每个单元有两个数, 第一个数是行变量和列变量之间的相关系数, 第二个数是检验这两个变量之间相关系数为0的检验的显著性概率值(p值)。

CORR的结果也可以在INSIGHT中由“Analyze - Multivariate”菜单得到。

3.8 用SAS/GRAPH绘图

SAS可以把存贮在SAS数据集中的数据以图形的方式形象直观地显示出来。在SAS / GRAPH 模块的支持下, SAS 可以作散点图、曲线图、直方图、扇面图、三维曲面图、等高线图、地图, 等等。

3.8.1 散点图和曲线图

用GPLOT 过程绘制散点图和曲线图。比如, 我们要绘制SASUSER. GPA 中SATV 对SASM 的散点图, 只要用此程序:

```
proc gplot data=sasuser.gpa;
  symbol i=none v=star;
```

```
  plot satv*satm;
run;
```

结果显示了一个GRAPHICS窗口, 绘出了以SATV为纵轴、以SATM为横轴的散点图(见图3.5)。在GPLOT过程中, 用PLOT语句指定绘图用的变量。SYMBOL语句是一个全程语句, 指定绘图用的连线方式、颜色、散点符号、大小, 等等。SYMBOL语句可以带编号, 如SYMBOL2, SYMBOL3等, 不带编号的相当于SYMBOL1。

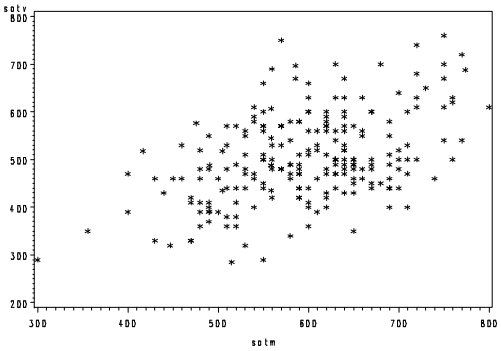


图 3.5: SATV对SATM的散点图

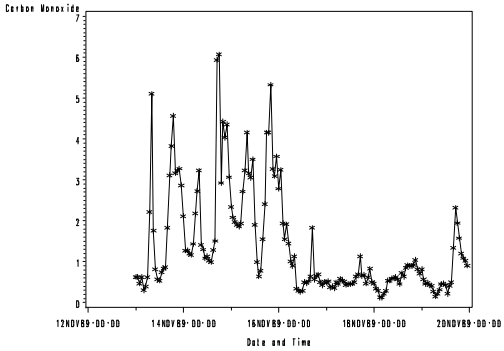


图 3.6: 一氧化碳浓度的时间序列曲线图

为了绘制连线, 只要在SYMBOL语句中指定I=JOIN。比如, 对SASUSER. AIR数据集, 以DATETIME为横轴、以CO为纵轴绘曲线图, 可以用:

```
proc gplot data=sasuser.air;
  symbol i=join v=star;
  plot co*datetime;
run;
```

见图3.6。如果不想在图中出现散点符号可以在SYMBOL语句中用V = NONE。

为了在图中作几条曲线, 只要在PLOT语句中指定多个因变量(自变量一般应为同一个), 并使用OVERLAY选项, 如：

```
proc gplot data=sasuser.air;
  symbol1 color=black i=join v=none line=1 ;
  symbol2 color=blue i=join v=none line=2 ;
  plot co*datetime=1 so2*datetime=2 / overlay;
run;
```

其中我们指定了两个SYMBOL语句, 第一个SYMBOL语句指定了LINE=1, 表示线型为实线, 第二个SYMBOL 语句指定了LINE=2表示线型为虚线。我们在PLOT语句中用了“纵轴*横轴=n”的格式来指定曲线使用哪一个SYMBOL 语句的规定来画, n对应于SYMBOL语句的序号。见图3.7。

SYMBOL语句的I=选项还可以取SPLINE表示在散点间连接连样条曲线, 取I=SMnn(nn取00－99值)表示绘制样条曲线但可以不经过散点, nn值代表曲线光滑性与拟和度的折中。取I=NEEDLE绘制每个点到横轴的垂线。取I=RL绘制线性回归直线, I=RQ为二次曲线, I=RC为三次曲线, 后面加上CLInn如RLCLI95表示在回归直线之外绘制预测值的95%置信限曲线, 比如：

```
proc gplot data=sasuser.gpa;
  symbol i=rlcli95 v=star;
  plot satv*satm;
run;
```

见图3.8。GPLOT过程还可以有其它灵活的用法, 可以参考有关资料或用SAS系统菜单“Help - SAS System Help”, 从中找到“Help on SAS Software Products - Using SAS / GRAPH Software”。

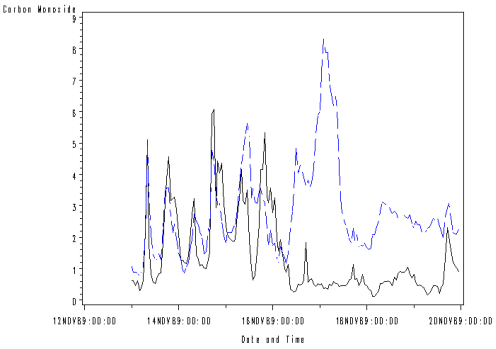


图 3.7: 两条曲线的例子

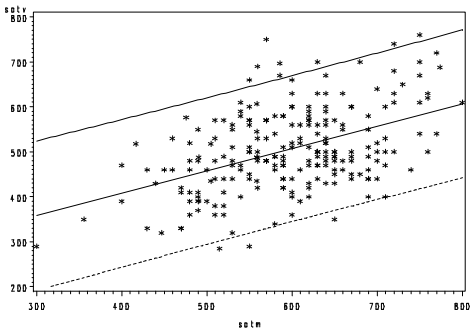


图 3.8: 带有95%预测限的回归直线和散点

3.8.2 直方图和扇形图

用GCHART过程绘制直方图、扇形图、三维直方图等表示变量分布的图形。例如, 要绘制SASUSER.GPA中GPA的分布直方图, 只要用:

```
proc gchart data=sasuser.gpa;
  vbar gpa;
run;
```

其中绘图用的变量用VBAR语句给出。图形见图3.9。如果把VBAR 改成HBAR 则条形方向变为横向。用GCHART 绘制的直方图和在INSIGHT 中绘制的直方图有所不同, 它在横轴标的是区间的中点值, 而在INSIGHT 中横轴标的是区间的端点值。

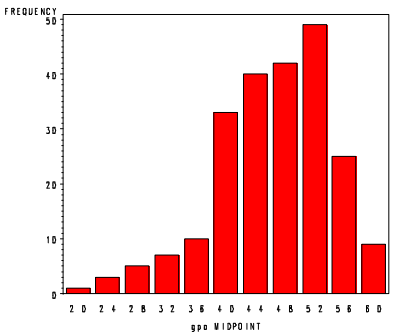


图 3.9: GPA的直方图

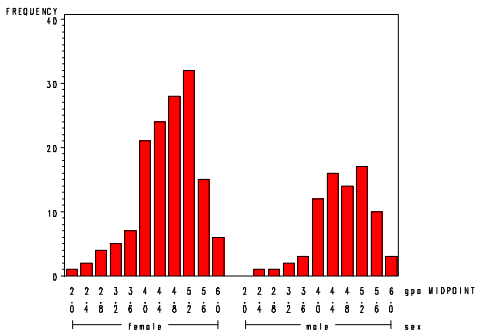


图 3.10: 并排直方图

可以绘制分组的直方图, 例如按性别分组绘制两个直方图并排放置, 可以用如下程序：

```
proc gchart data=sasuser.gpa;
  vbar gpa / group=sex;
run;
```

结果见图3.10。

在GCHART中用PIE语句绘制表示频数的扇形图, 例如：

```
proc gchart data=sasuser.gpa;
  pie sex;
run;
```

结果见图3.11。如果想显示百分比值, 只要在PIE语句中加入TYPE = PERCENT选项, 如“pie sex / type=percent;”。

GCHART过程还可以用BLOCK 语句绘制三维直方图。例如, 在SASUSER. HOUSES有各种房子的情况, 其中BEDROOMS 表示卧室个数, STYLE 表示房子的式样, 都是分类变量, 我们可以用如下程序画

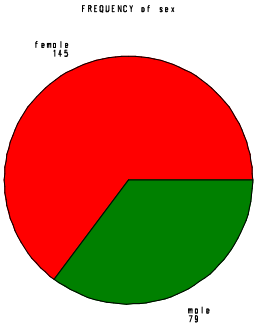


图 3.11: 扇形图

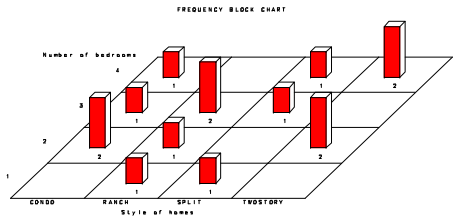


图 3.12: 三维直方图

出反映卧室个数和房子式样的联合分布的三维直方图：

```
goptions hpos=90 vpos=70;
proc gchart data=sasuser.houses;
  block style / group=bedrooms;
run;
```

图形见图3.12。

3.8.3 三维曲面图和等高线图

假设对一个二元函数z=f(x,y), 我们有了x取等间隔值、y取等间隔值时z的值, 这时我们可以用G3D 过程绘制曲面图形, 用GCONTOUR 绘制曲面的等高线图。

例如, 我们想绘制一个二维正态分布曲面的图形, 假设(X, Y)服从联合正态分布, 其均值都是0, 方差分别为1和a, 相关系数为r。这时, 我们可以得到(X, Y)的联合密度函数的公式为：

$$f(x,y)=\frac{1}{2\pi a\left(1-r^2\right)}\exp\left(-\frac{1}{2a\left(1-r^2\right)}\left(ax^2+y^2-2r\sqrt{a}xy\right)\right)$$

我们可以在一个网格上计算曲面的值：

```
data dnorm2;
  a=2;
  a2=sqrt(a);
  r=0.5;
  det=a*(1-r*r);
  do x=-3 to 3 by 0.3;
    do y=-3*a2 to 3*a2 by 0.3*a2;
      z=1/(2*3.1415926*det)*exp(-0.5/det*
        (a*x*x + y*y - 2*r*a2*x*y));
      output;
    end;
  end;
  keep x y z;
run;
```

然后, 我们就可以用G3D过程来绘制曲面图：

```
proc g3d data=dnorm2;
  plot x*y=z;
run;
```

见图3.13。用GCONTOUR 过程可以绘制曲面对应的等高线图, 例如：

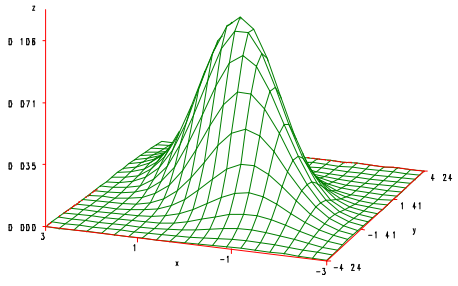


图 3.13: 曲面图

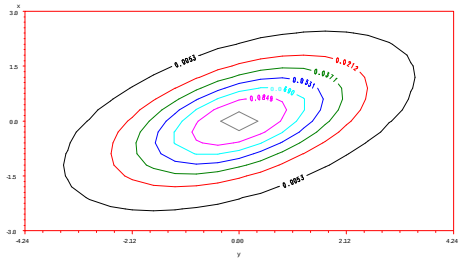


图 3.14: 等值线图

```
proc gcontour data=dnorm2;
  plot x*y=z / nolegend autolabel;
run;
```

见图3.14。

在INSIGHT 中画旋转图时在Output 选项中选中Fit surface 可以作曲面图；用“Analyze - Contour Plot(Z Y X)” 菜单可以作等值线图。

3.8.4 图形的调整与输出

各绘图过程中都指定了丰富的选项来调整图形, 读者可以参考有关资料或查系统的帮助。另外, 在图形中也可以用TITLE语句和FOOTNOTE语句给图形加标题和脚注。

为了在图形的标题、标签中也能使用汉字, 老的SAS版本需要比较多的步骤, 但是在Windows下的SAS 8.0以后只要很简单的语句就可以在图形中使用True Type字型的汉字, 例如：

```
goptions ftext="宋体" htitle=2 cells htext=1 cells;
proc gplot data=sasuser.class;
  title "试验SAS图形的汉字功能";
  symbol i=none v=square;
  plot weight * height;
  label weight = "体重" height="身高";
run;
```

见图3.15。其中GOPTIONS中的FTEXT选项指定图形中文本的字体。如果需要用汉字则需要指定一种汉字字体。HTITLE选项和HTEXT选项指定标题和其他文字的高度, 如无需要可以省略。

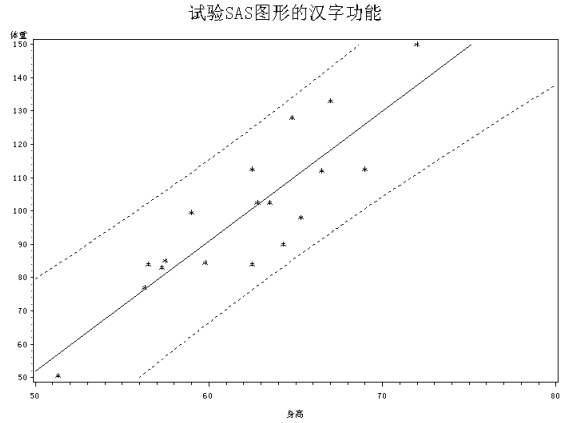


图 3.15: 在SAS图形中使用汉字

为了把SAS/GRAPH绘制的图形保存为兼容的图形文件, 只要在显示某一页图形输出时, 选“File - Export”菜单, 在出现的输入文件名的窗口选择适当的文件类型, 如BMP、WMF、JPEG、PS, 等等。本讲义中的SAS/GRAPH图形就是从SAS中用“File - Export”输出为矢量图格式的EMF文件再处理的结果。

要打印SAS/GRAPH生成的图形, 只要选“File - Print”。这样用Windows的打印驱动程序与SAS/GRAPH的图形驱动配合来打印。另外, SAS/GRAPH模块还提供了许多种打印机的独立的驱动程序, 可以不依赖于Windows的打印驱动, 具体请参考有关资料或帮助。

3.9 分析员模块介绍

Analyst (分析员)是用图形界面调用SAS 功能的一个模块，它可以管理数据，计算描述统计量，进行假设检验，做回归分析和方差分析，作各种图

形。Analyst 是用数据步编程和调用SAS 过程联合完成任务的，所用的SAS 程序还可以显示在一个单独的窗口供我们学习参考。这里介绍其数据管理、描述统计、作图等功能。

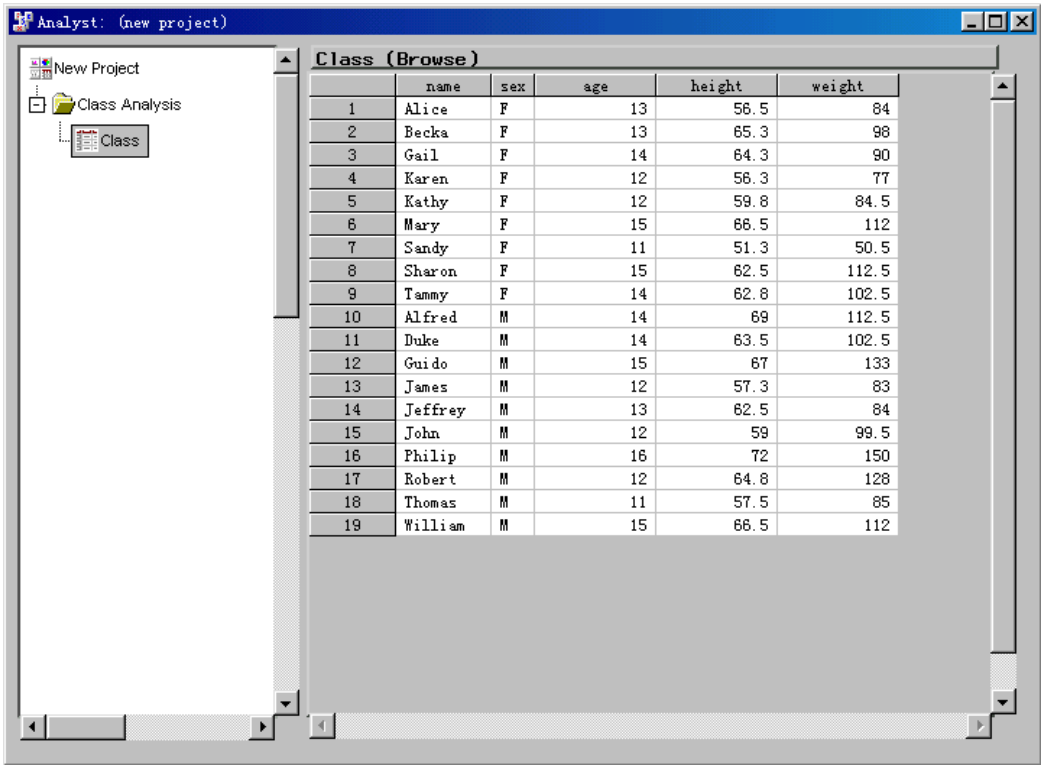
从主菜单“Solutions - Analysis - Analyst” 可以进入Analyst窗口。系统自动生成一个新的分析项目。Analyst 窗口分为左右两部分，左半部分为项目管理，用树形目录显示打开的数据及对其所进行的分析；右半部分显示数据。输出结果的名字会列在项目管理的树形目录中，结果的文本或图形会显示在单独的窗口中。

3.9.1 数据管理

Analyst 也有一个数据窗口可以进行数据管理。比如，我们按如下方法打开SASUSER.CLASS: 选“File - Open by SAS Name”, 选择SASUSER 库, 再选其中的CLASS 数据集。见图3.16。Analyst也允许直接按文件名打开数据。

编辑

打开的数据缺省状态为只读浏览(Browse)。为了能编辑数据，需要调用菜单“Edit - Mode - Edit”。这时只要把光标单击到某一个数据格就可以修改这个格的内容。为了保存所作的修改需要调用“File - Save”菜单。单击变量名或行号可以选定变量或行。在变量名或行号上打开右键菜单中可以增加新列(Insert)或新行(Add)，可以复制选定的行或



	name	sex	age	height	weight
1	Alice	F	13	56.5	84
2	Becka	F	13	65.3	98
3	Gail	F	14	64.3	90
4	Karen	F	12	56.3	77
5	Kathy	F	12	59.8	84.5
6	Mary	F	15	66.5	112
7	Sandy	F	11	51.3	50.5
8	Sharon	F	15	62.5	112.5
9	Tammy	F	14	62.8	102.5
10	Alfred	M	14	69	112.5
11	Duke	M	14	63.5	102.5
12	Guido	M	15	67	133
13	James	M	12	57.3	83
14	Jeffrey	M	13	62.5	84
15	John	M	12	59	99.5
16	Philip	M	16	72	150
17	Robert	M	12	64.8	128
18	Thomas	M	11	57.5	85
19	William	M	15	66.5	112

图 3.16: Analyst界面

列(Duplicate)，可以删除选定的行或列(Delete)。在变量名上打开右键菜单可以改变变量的次序(Move)，隐藏某些变量使其不参加分析(Hide)，取消隐藏(Unhide)，指定到本列为止的列在左右翻看时不动(Hold)，按变量值排序(Sort)，切换显示变量标签(Labels)，查看变量属性(Properties)。

数据筛选

Analyst 的Data 菜单提供了对数据的一些操作。本小节和以下几小节介绍这些操作。

Data - Filter - Subset Data 菜单可以用Where

表达式选择满足条件的行子集，随后的分析只对选定的子集进行，可以用“File - Save As”菜单来保存筛选出的子集。注意，“File - Save”菜单仍保存全集。

数据变换

Data - Transform 菜单提供了一些按程序修改变量的办法。这些修改也必须在编辑状态才能启用。

Compute 可以构造一个表达式来生成新的变量，不过这里用图形界面构造表达式的办法实在还不如直接用数据步编程简单。

Rank 可以计算某变量的秩统计量加入到数据集中，在计算秩统计量的对话框中还可以规定升序、降序，可以指定分组变量以在组内排序，可以用Options 钮打开一个对话框选择计算秩统计量的具体方法。

Standardize 可以把一个变量标准化为均值零、标准差等于1。

Recode Values 可以把变量的每个取值用指定的值代替从而生成一个新的重编码的变量，选此菜单后选择要重编码的变量，要生成的变量名及类型，然后出现一个界面显示变量的所有不同值让用户输入要改成的新值。

Recode Ranges 把一个变量分段然后把每一段对应到一个编码从而生成一个新的重编码的变量，选此菜单后选择要重编码的变量，要生成

的变量名及类型，要分的组数，然后出现一个对应表要求用户输入每一组的范围和新编码。

Change Type 可以通过转换变量的数值型和字符型生成新变量，如果输入数据时错误地把数值型列当成了字符型列来输入的情况可以用这个功能来弥补。

Log(Y)等 选定某列后可以通过计算此列的常见函数来生成新变量，这些函数包括自然对数、平方根、倒数、平方、指数函数。

随机数生成

Data - Random Variates 菜单可以生成一列随机数。支持的分布类型有：正态、均匀、二项、卡方、泊松、贝塔、指数、伽马、几何、极值分布。选择需要的分布后输入分布参数就可以生成一个新变量保存生成的一列随机数。

分类汇总

Summarize By Group 菜单提供了一个方便的数据汇总功能，可以计算分类的统计量。比如，SASHELP.PRDSALE 是一个很大的销售数据集，直接查看比较困难，我们就可以按几个分类对其进行汇总。比如，我们计算按照产品类型(PRODTYPE)和产品名称(PRODUCT)分类的总销售额、最高、最低，可以选"Data - Summarize By Group"，出现如图3.17的对话框，选定销售额ACTUAL，按Summarize 钮指定它为

要汇总的变量，选定PRODTYPE 和PRODUCT按Group钮指定用这两个变量交叉分组，选统计量Sum(总和)、Minimum(最低)、Maximum(最高)，确定(OK)后就得到了分类汇总后的数据集，显示在一个单独的数据窗口，可以用“File - Save as By SAS Name”保存为数据集。

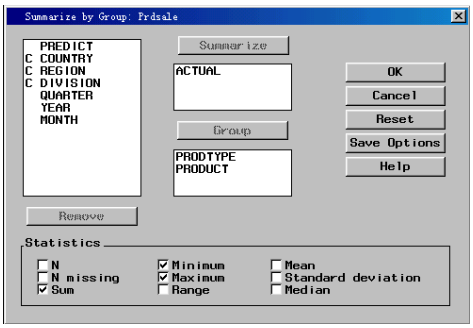


图 3.17: Analyst: 分类汇总

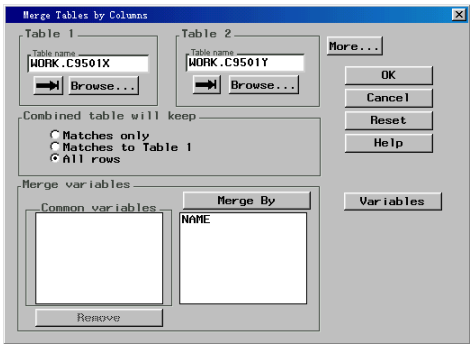


图 3.18: 横向合并

横向合并

Data - Combine Tables - Merge By Columns 菜单相当于数据步的MERGE 语句。可以横向合并两个数据集。例如要合并2.3.8中的C9501X 和C9501Y，可以选“Data - Combine Tables - Merge By Columns”菜单，出现如图3.18的对话框，选Table 1 和Table 2 分别为WORK.C9501X 和WORK.C9501Y，这时在Common variables框中将出现NAME，这是两个数据集中都有的变量，可以按它合并数据集，选NAME 再按Merge By钮把它移到Merge By框就指定了按姓名合并。对话框中的Combine table will keep有三个选择：Matches only 指两个数据集都出现同

一个名字才保留合并的行；Matches to Table 1 指只要在第一个数据集中出现的名字就予以保留；All rows 指两个数据集中出现的名字不管能否匹配都要保留(这是数据步中MERGE 语句的缺省操作)。这些功能是通过数据步编程完成的，可以选Analyst 左边的项目管理树中的Code 条目来查看使用的数据步程序。合并对话框中的Variables 按钮可以指定最后保留的变量。

纵向合并

Data - Combine Tables - Concatenate By Rows 菜单相当于数据步中用SET 纵向合并数据集。合并有两种选择：Append 是直接上下合并，这是SET 的缺省操作；Interleave(交错) 是合并时按照某个变量的值合并，该变量值相同的观测相邻放置，确保最后得到的数据集按此变量排序。我们第二章讲纵向合并时没有讲交错合并，读者可以查看Analyst 生成的代码来学习这种作法。

行列转置

Data - Stack Columns 菜单堆叠若干列。比如，下面的数据步生成了10个观测，每个观测中包含序号I和三个重复试验Y1、Y2、Y3 三列:

```
data tran;
  array y(3);
  do i=1 to 10;
     do j=1 to 3;
        y(j) = uniform(0);
     end;
```

```
    output;
  end;
  drop j;
run;
```

为了把这三个变量合并为一个变量Y的三次观测，就可以调用这个菜单，在弹出的对话框中选Y1, Y2, Y3为Stack 变量(要把这三个列堆起来成为三行)，堆叠后的列名(Stacked column)为Y, 来源列名(Source column)为_SOURCE_ (这个列保存被堆叠的三个列的名字), 就可以把Y1, Y2, Y3堆叠起来，这样原来的10行变成了30行，每个I的值有三个观测。用File 菜单可以保存结果，假设保存到了数据集TRAN_ST。

Data - Split Columns 菜单指定一个分组变量和一个要拆分的列，按分组变量的变化把拆分列中的各观测拆成与分组变量对应的若干列。这基本是上面的堆叠操作的逆。比如要把上面得到的TRAN_ST中的Y仍拆为三个，就可以选此菜单，在弹出的对话框中选要分的变量(Split Column)为Y，分组依据(Group By)为_SOURCE_，就可以拆分得到新数据集。

Data - Transpose 菜单对数据集进行转置，原来的行变成列，列变成行。在转置时可以指定一个分组变量，这时每一个转置只在一个分组内部进行，分组变量本身并不改变。前面的列堆叠和拆分基本是这种转置的特例。比如，要把数据集TRAN中的Y1, Y2, Y3合并为一个变量的三个观测，只要调用这个转置菜单，选Y1, Y2, Y3为要转置的变量，

选I为分组变量，则可以达到把Y1, Y2, Y3合并的效果。反之，为了把TRAN_ST的变量Y重新拆分成三列，只要调用转置菜单，选Y为要转置的变量，选I为分组变量即可。

随机抽样

Data - Random Sample 菜单对数据集的观测进行随机抽样。因为数据集的每行是一个观测，我们可以把这些观测看成是一个有限总体进行抽样。选此菜单后将显示原始数据集的行数，用户可以选择要抽取的样本量(Rows)或抽取的百分比(Ratio)，可以指定随机抽样所用的随机数种子。

3.9.2 报表

Analyst 支持PRINT 过程的列表和TABULATE 过程的汇总表。

列表

选“Reports - List Data”菜单弹出如图3.19的对话框，这里的选项基本与PRINT 过程对应。Print 框中是要显示的变量。Id框中相当于ID 语句指定的识别变量。按Titles 按钮出现一个设定标题的对话框，可以设定全局标题、只用于本过程的标题、是否包括日期时间、页号、观测筛选条件等。按Options 按钮可以设定PRINT 过程的选项，包括是否使用变量标签作为列标题，列表行距宽窄，是否列出行号，是否

列出总观测个数，可以指定对某些变量求总计(相当于SUM 语句)。

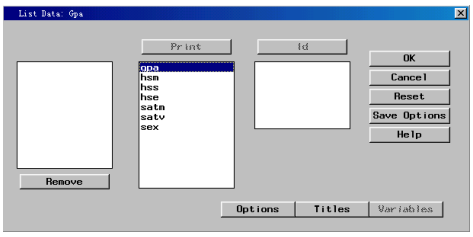


图 3.19: Analyst: Analyst: 列表报告

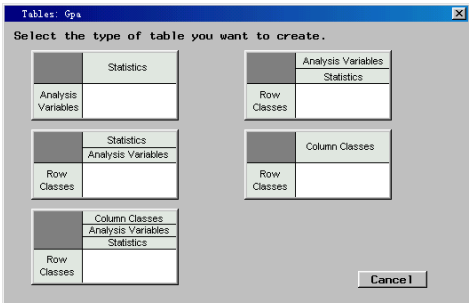


图 3.20: Analyst: 五种汇总表

汇总表

选“Reports - Tables”菜单可以弹出如图3.20的选择，这是Analyst 用来制作五种常见汇总表的一个简单易用的界面。这个界面用起来比写程序还要方便，我们举例说明。比如对GPA数据集我们希望分男女计算SATM和SATV的平均值、标准差、中位数，可以按如下步骤：按图例的第三种，弹出如图3.21的对话框。在这里我们可以指定分析变量(Analysis Variables)为satm 和satv，行分类变量(Row Classes)为sex, 统计量为MEAN, STD 和MEDIAN，再按Options窗口弹出如图3.22的对话框，这里的General 部分可以指定是否计算合计，缺省数据输出格式，Labels 部分可以指定变量和统计量的标签。

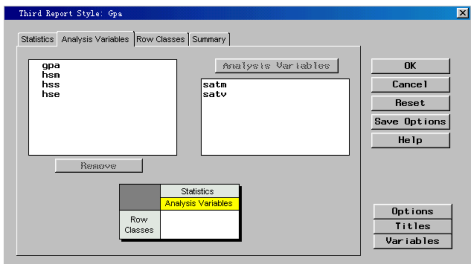


图 3.21: Analyst: Analyst: 第三种汇总表

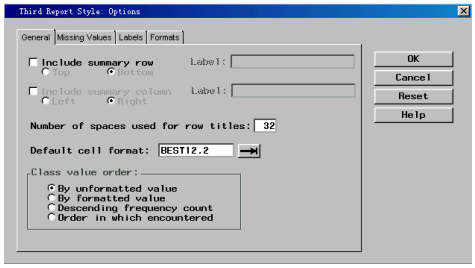


图 3.22: Analyst: 汇总表选项

3.9.3 描述统计

Analyst 的“Statistics - Descriptive”菜单可以计算描述统计量。其中的“Summary Statistics” 相当于MEANS过程, 调用此菜单后弹出如图3.23的对话框。这里我们可指定分析变量(Analysis)，用Statistics 的选项可以指定要计算的统计量(包括所有的矩统计量), 用Plots 的选项可以要求绘制变量的盒形图和直方图，用Output 的选项可以指定数值输出格式和是否使用变量标签，用Variables 的选项可以指定分组变量(By Group)、权重变量(Weight)和频数变量(Frequency)，用Save Data 的选项把统计量写到一个SAS数据集。

“Statistics - Descriptive - Distributions”相当于UNIVARIATE 过程，但是在Analyst 中又通过编程提供了增强的功能。选此菜单后弹出的对话框如图3.24。我们首先要选定分析变量(Analysis)，用Method 选项选择计算样本方差时使用的分母，用Fit选项可以要求对数据拟合正态、对数正态、指数、威布尔等四种分布，用Plots可以要求绘制盒

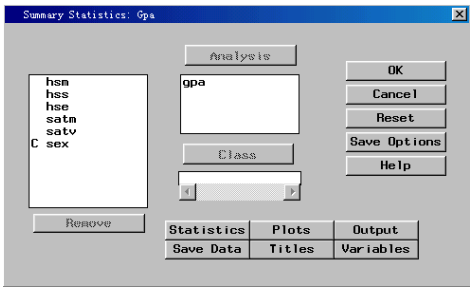


图 3.23: Analyst: Analyst: 基本统计量

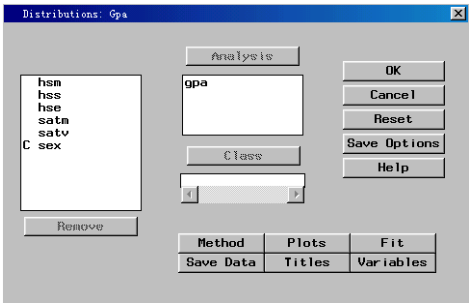


图 3.24: Analyst: Analyst: 分布研究

形图、直方图、概率图和QQ图等四种图形，如果已经要求了拟合分布则直方图中将叠加拟合的分布密度，概率图和QQ图也是相对于要拟合的分布来画。概率图(Probability Plot)和QQ图基本相同，两者纵坐标都是分析变量由小到大排序后的数值y_i，若y_i相当于v_i经验分位数(可以是修正过的)，这时正态QQ图横坐标是标准正态的v_i分位数，横坐标的标度也是正态分位数；而正态概率图的横坐标仍用标准正态分位数但是标度却标分位数相应的概率值即v_i本身。

“Statistics - Descriptive - Correlations” 菜单计算变量的相关系数，相当于CORR 过程，这里增强的地方是可以画散点图并在散点图上画置信椭圆。

“Statistics - Descriptive - Frequency Counts” 菜单进行变量分布频数统计，相当于FREQ 过程，同时还可以画离散变量的条形图。

3.9.4 画图

用Analyst 的Graphs 菜单可以很容易地作出数据集中

变量的各种图形。

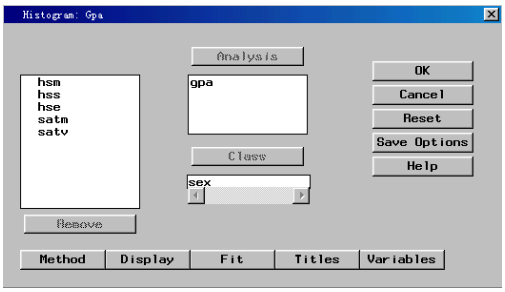


图 3.25: Analyst: 直方图

“Graphs - Bar Chart” 画条形图。可以画通常的竖立的条形图，也可以画水平的条形图。可以把条形画成三维形状。区间变量的条形图与直方图没有本质区别，但是分类变量只能作条形图不能作直方图。

“Graphs - Pie Chart” 画扇形图，可以画成三维的形状。

“Graphs - Histogram” 画直方图。Analyst 的这个菜单有增强的功能。选此菜单后弹出的对话框如图3.25所示。我们需要指定分析变量(Analysis)。通过Fit的选项可以要求拟合正态、对数正态、指数、威布尔四种分布并把拟合的分布密度曲线画在直方图上面。

“Graphs - Box Plot” 画盒形图。设定盒形图的对话框中有一个Display 按钮可以选择具体画法，我们通常见的盒形图在这里相当于Style 为Schematic。

“Graphs - Probability Plot” 画概率图，以数据集中的变量值为纵坐标，以拟合的理论分布的对应分位数为横坐标但横坐标轴标的是概率值。可以选正态分布、对数正态分布、指数分布、威布尔分布。

“Graphs - Scatter Plot” 画二维散点图或三维散点图。

“Graphs - Contour Plot” 关于X, Y, Z 三个坐标画等

值线图。

“Graphs - Surface Plot” 关于X, Y, Z 三个坐标画曲面图。

练习

1. SASHELP.PRDSALE是某国际公司在各地销售记录。变量ACTUAL是实际销售额, PREDICT 是预测的销售额，COUNTRY是卖往的国家, REGION是地区, DIVISION是卖往的部门, PRODTYPE是产品类型, PRODUCT是具体的产品名称, QUARTER, YEAR, MONTH是销售时间的季度, 年, 月。

   (1)列出数据集的内容, 要求给各列加上合理的中文列标题, 不要观测序号。

   (2) 把数据集按产品类别、产品、年、月排序后按产品类别、产品分类列出年、月、销售额，计算销售额的小计和总计。

   (3) 用TABULATE过程绘制按产品类别和产品名称交叉分类的销售额及总计。使用中文标签。

2. 对SASUSER.GPA中 的GPA用UNIVARIATE过 程分析分布并简述结果。叙述性别分布。

3. 绘制F分布自由度为(1,30), (2,30), (3,30), (4,30), (5,30), (10,30)的密度曲线图(画在同一坐标系中)。

第四章 SAS的基本统计分析功能

前面我们已经看到了SAS的编程计算、数据管理能力、数据汇总、数据探索分析能力。这一章我们讲如何用SAS进行基本的统计检验、线性回归、方差分析、列联表检验等基本统计分析。我们既使用SAS语言编程, 也使用SAS/INSIGHT 和Analyst 的菜单界面。

4.1 一些单变量检验问题

对单个变量, 我们可能需要作正态性检验、两独立样本均值相等的检验、成对样本均值相等的检验。

4.1.1 正态性检验

在PROC UNIVARIATE 语句中加上NORMAL 选项可以进行正态性检验。例如, 我们要检验SASUSER.GPA中GPA是否服从正态分布, 只要用如下UNIVARIATE过程：

```
proc univariate data=sasuser.gpa normal;
  var gpa;
run;
```

结果(部分)如下：

Tests for Normality				
Test	--Statistic---		-----p Value------	
Shapiro-Wilk	W	0.966294	Pr < W	<0.0001
Kolmogorov-Smirnov	D	0.059805	Pr > D	0.0488
Cramer-von Mises	W-Sq	0.212179	Pr > W-Sq	<0.0050
Anderson-Darling	A-Sq	1.564677	Pr > A-Sq	<0.0050

这里给出了GPA 变量的四种正态性检验结果, 其中Shapiro-Wilk检验是我们首选的。我们可以看到, p值很小, 所以在0.05水平(或0.10水平)下应拒绝零假设, 即认为GPA分布非正态。

在SAS / INSIGHT 中为了检验GPA 的分布, 先选“Analyze - Distribution”菜单打开GPA变量的分布窗口, 然后选“Curves - Test for Distribution” 菜单。除了可以检验是否正态分布外还可以检验是否对数正态、指数分布、Weibull分布。

在Analyst 中选“Statistics - Descriptive - Distributions” 调出分布研究对话框, 选了分析变量后按Fit钮可以要求进行分布拟合, 最后的结果中包含了分布拟合的三种检验(为上面Univariate的四种检验的后三种)。这里还可以要求画盒形图、直方图(要求拟合时直方图上面会附加拟合的正态密度曲线)、概率图、QQ图。Analyst的这个菜单也可以进行对数正态、指数和Weibull分布的拟合和检验。

4.1.2 两独立样本的均值检验

假设我们有两组样本分别来自两个独立总体, 需要检验两个总体的均值或中心位置是否一样。如果两个

总体都分别服从正态分布, 而且方差相等, 可以使用两样本t检验过程TTEST。

比如, 我们要检验SASUSER.GPA数据集中男生和女生的SATM分数是否具有相等的平均值, 只要用如下程序：

```
proc ttest data=sasuser.gpa;
  class sex;
  var satm;
run;
```

过程中用CLASS语句指定分组变量, 用VAR语句指定要比较的变量。结果如下：

```
                    The TTEST Procedure

                         Statistics

                                    Lower CL            Upper CL
Variable   sex              N        Mean     Mean        Mean

satm       female         145      597.98   611.77      625.56
satm       male            79      546.45   565.03       583.6
satm       Diff (1-2)              23.698   46.747      69.796

                        Statistics

                      Lower CL             Upper CL
Variable   sex           Std Dev   Std Dev    Std Dev   Std Err

satm       female         75.336    84.021     94.986    6.9775
satm       male            71.71    82.929     98.343    9.3303
satm       Diff (1-2)      76.53    83.639     92.215    11.696

                        Statistics

         Variable   sex          Minimum    Maximum

         satm       female           400        800
         satm       male             300        740
         satm       Diff (1-2)
```

T-Tests

Variable	Method	Variances	DF	t Value	Pr > |t|
satm	Pooled	Equal	222	4.00	<.0001
satm	Satterthwaite	Unequal	162	4.01	<.0001

Equality of Variances

Variable	Method	Num DF	Den DF	F Value	Pr > F
satm	Folded F	144	78	1.03	0.9114

结果有三个部分：两个总体的SATM简单统计量(Statistics 部分, 包括变量分组的均值、标准差、两组平均值差、平均值差的标准差, 等等), 两样本均值的检验(T-Tests部分), 以及两样本方差是否相等的检验(Equality of Variances 部分)。标准的两样本t检验要求两总体方差相等, 所以第三部分结果检验两样本方差是否相等。如果检验的结果为相等, 则可使用精确的两样本t检验(方法为Pooled), 看第二部分结果中标Equal那一行。如果方差检验的结果为不等, 则只能使用近似的两样本t检验, 看第二部分结果的Unequal那一行(方法为Satterthwaite)。这里我们看到方差检验的p值为0.9114不显著, 所以可以认为方差相等, 所以我们看Equal行, p值为0.0001在0.05水平下是显著的, 所以应认为男、女生的SATM分数有显著差异, 女生分数要高。这里我们也可以不管方差是否相等都只使用Satterthwaite 方法的结果。

上面的检验中对立假设是两组的均值不等, 所以检验是双边的, 假设T_{n-1}是$n-1$个自由度的t分布随机变量, A是我们得到的T统计量的值, 则p值的计算

公式为$p = \Pr(|T_{n-1}| > |A|)$。如果要进行单边的检验, 比如对立假设为女生分数高于男生分数(右边), 则p值为$p = \Pr(T_{n-1} > A)$, 当计算得到的t统计量值为正数时(现在t=4.00)此单边p值为双边p值的一半, 当计算得到的t统计量为负数时肯定不能否定零假设。检验左边时恰好相反。

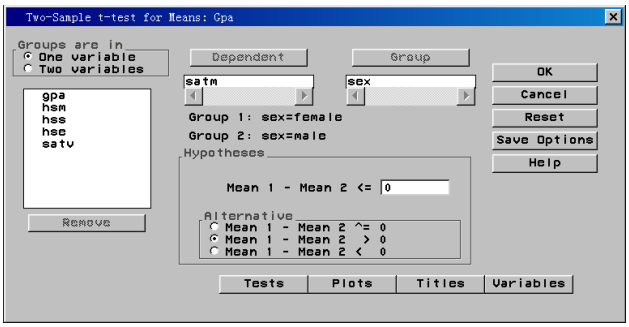


图 4.1: Analyst的两样本t检验对话框

SAS / INSIGHT未提供独立两样本均值检验的功能。Analyst中可以进行两样本t检验。从菜单“Statistics - Hypothesis Tests - Two-Sample t-test for Means” 打开检验的对话框如图4.1。这里我们除了可以进行双边检验外还可以选择单边检验(这一点相对于原始的TTEST 过程是一个进步), 比如选择对立假设为女生比男生SATM 分数好, 结果如下:

```
   Two Sample t-test for the Means of satm within sex      4


Sample Statistics

     Group          N      Mean    Std. Dev.   Std. Error
     ------------------------------------------------------
     female       145  611.7724      84.021       6.9775
     male          79  565.0253      82.929       9.3303
```

```
Hypothesis Test

     Null hypothesis:    Mean 1 - Mean 2 <= 0
     Alternative:        Mean 1 - Mean 2 >  0

     If Variances Are    t statistic      Df       Pr > t
     ---------------------------------------------------
     Equal                     3.997       222       <.0001
     Not Equal                 4.012    162.17       <.0001
```

可见女生的SATM 分数比男生高。

如果我们希望检验男、女生的GPA分数则无法使用两样本t检验, 因为检验女生的GPA样本的正态性发现它非正态。这种情况下我们可以使用非参数检验。检验两独立样本的中心位置是否相同的非参数检验有Wilcoxon秩和检验。我们用NPAR1WAY过程加Wilcoxon选项可以进行这种检验。见下例：

```
proc npar1way data=sasuser.gpa wilcoxon;
  class sex;
  var gpa;
run;
```

其CLASS语句和VAR与TTEST过程相同。结果如下：

```
                    The NPAR1WAY Procedure

          Wilcoxon Scores (Rank Sums) for Variable gpa
                   Classified by Variable sex

                    Sum of    Expected     Std Dev        Mean
  sex        N      Scores    Under H0    Under H0       Score
  ------------------------------------------------------------
  female   145    16067.50    16312.50  463.429146  110.810345
  male      79     9132.50     8887.50  463.429146  115.601266

              Average scores were used for ties.
```

```
                Wilcoxon Two-Sample Test

          Statistic              9132.5000

          Normal Approximation
          Z                         0.5276
          One-Sided Pr >  Z         0.2989
          Two-Sided Pr > |Z|        0.5978

          t Approximation
          One-Sided Pr >  Z         0.2992
          Two-Sided Pr > |Z|        0.5983

     Z includes a continuity correction of 0.5.


           Kruskal-Wallis Test

          Chi-Square                0.2795
          DF                             1
          Pr > Chi-Square           0.5970
```

结果分为四部分：两样本的秩和的有关统计量, Wilcoxon 两样本检验的结果, t检验的近似显著性, Kruskal-wallis 检验结果。我们只要看Wilcoxon 检验的p值Prob $>|$ Z $|=0.5978$, 检验结果不显著, 可认为男、女生的GPA分数在0.05水平下无显著差异。这里还给了单边检验的结果, 单边检验的对立假设是男生分数比女生高, 结果也不显著。

Analyst中也提供了Wilcoxon 秩和检验。从“Statistics - ANOVA - Nonparametric One-Way ANOVA”菜单进入, 它也是使用NPAR1WAY 过程。结果与上述结果有细微的差别, 这是计算Z统计量时作连续性修正的影响引起的。

4.1.3 成对总体均值检验

我们在现实中经常遇到同一总体两个测量结果的比

较, 比如, 考察同一组人在参加一年的长跑锻炼前后的心率有无显著差异。这时, 每个人一年前的心率和一年后的心率是相关的, 心率本来较快的人锻炼后仍相对于其它人较快。所以, 检验这样的成对总体的均值不能使用两样本t检验的方法, 因为独立性条件不再满足。这时, 我们可以检验两个变量间的差值的均值是否为零, 这等价于检验两组测量值的平均水平有无显著差异。

检验单个样本的均值是否为零只要使用UNIVARIATE过程, 在UNIVARIATE过程的矩部分给出了均值为零的t检验和符号检验、符号秩检验的结果。例如, 我们想知道SATM和SATV这两门考试的成绩有无显著差异(SATM平均值为595.3, SASTV平均值为504.6, 我们希望知道差异是否显著)。因为这两个成绩是同一个学生的成绩, 所以它们之间是相关的(学得好的学生两科一般都好, 学得差的一般两科都差), 不能用独立两样本的t检验, 但可以计算两变量间的差DMV=SATM−SATV, 检验差值变量的均值是否为零。如果否定, 则可认为SATM和SATV的平均值有显著差异。

为此, 我们先用一个数据步计算差值, 然后对差值变量用UNIVARIATE过程进行分析就可以得到结果。程序如下：

```
data new;
  set sasuser.gpa;
  dmv = satm - satv;
  keep dmv;
run;
proc univariate data=new;
  var dmv;
```

```
run;
```

结果(部分)如下：

```
              Basic Statistical Measures

    Location                    Variability

Mean     90.73661     Std Deviation           92.82931
Median   90.00000     Variance                    8617
Mode     90.00000     Range                   490.00000
                      Interquartile Range     130.00000


              Tests for Location: Mu0=0

   Test           -Statistic-    -----p Value------

   Student's t    t  14.62923    Pr > |t|    <.0001
   Sign           M      73.5    Pr >= |M|   <.0001
   Signed Rank    S    9757.5    Pr >= |S|   <.0001
```

其中的位置检验(Tests for Location: Mu0=0) 部分是假设检验问题$H_0 : \mu = 0 \longleftrightarrow H_a : \mu \neq 0$的检验结果。第一个检验为t检验(Student's t), 需要假定差值变量服从正态分布, 检验的p值Pr> |t| < .0001, 这个检验在0.05水平下是显著的, 所以可认为两科分数有显著差异。第二个检验(Sign)是叫做符号检验的的非参数检验, 其p值为Pr>= |M| <.0001, 在0.05水平下也是显著的, 结论不变。第三个检验(Sgned Rank)是叫做符号秩检验的非参数检验, 其p值为Pr>= |S| < .0001, 在0.05水平下是显著的, 结论不变。所以这三个检验的结论都是两科成绩有显著差异。

如果t检验对立假设是单边的, 其p值算法与上面讲的两样本t检验p值算法相同。

在SAS/INSIGHT中比较成对样本均值的显著差异, 同样是先计算两变量的差值变量(在“Edit - Variables - Other”菜单中, 指定两个变量, 指定两个变量间的计算为减法, 则可以生成差值变量, 可以用数据窗口菜单的“Define Variables”改变量名), 然后对此差值变量选“Analyze - Distribution”, 选“Tables - Tests for Location”就可以在分布窗口显示这三个检验的结果。

Analyst 中成对t检验十分方便, 不需要自己计算差值变量, 只要选菜单“Statistics - Hypothesis Tests - Two-Sample Paired t-test for Means”, 在弹出的对话框中给出第一组变量名SATM, 第二组变量名SATV，就可以进行成对t检验。这里除了可以作双边检验以外还可以作单边检验，见图4.2。

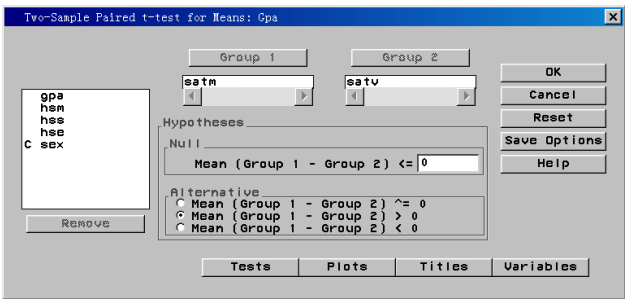


图 4.2: Analyst的成对t检验对话框

4.2 回归分析

本节先讲述如何用SAS/INSIGHT进行曲线拟合, 然后进一步讲如何用SAS/INSIGHT进行线性回归, 简单介绍SAS/INSIGHT的广义线性模型拟合, 最后介绍如何用编程和Analyst 进行回归分析。

4.2.1 用SAS/INSIGHT进行曲线拟合

两个变量Y和X之间的相关关系经常可以用一个函数来表示, 一元函数可以等同于一条曲线, 实际工作中经常对两个变量拟合一条曲线来近似它们的相关关系。最基本的“曲线”是直线, 还可以用多项式、样条函数、核估计和局部多项式估计曲线。其模型可表示为

$$Y = f(X) + \varepsilon$$

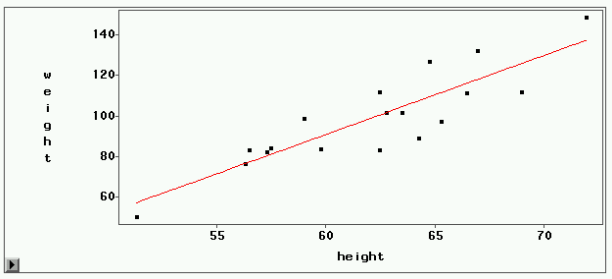


图 4.3: 身高对体重的散点图及回归直线

例如, 我们要研究SASUSER.CLASS数据集中学生体重与身高之间的相关关系。为此, 我们可以先画出两者的散点图(Analyze - Scatter plot)。从图中可以看出, 身高越高的人一般体重越重。我们可以把体重作为因变量、身高作为自变量拟合一条回归直线, 只要选“Analyze - Fit (Y X)”, 并选体重为Y变量, 身高为X变量, 即可自动拟合出一条回归直线, 见4.3。窗口中还给出了拟合的模型方程、参数估计、诊断信息等, 我们在下一小节再详细介绍。

在拟合了直线后, 为拟合多项式曲线, 只要选“Curves - Polynomial”, 然后输入阶

次(Degree(Polynomial)), 就可以在散点图基础上再加入一条多项式曲线。对于本例, 我们看到二次多项式得到的曲线与直线差别很小, 所以用二次多项式拟合没有优势。还可以试用三次、四次等多项式。为了改变阶次还可以使用拟合窗口中的多项式阶次滑块(Parametric Regression Fit中的Degree(Polynomial))。这里我们试着增大多项式阶可以发现取太高阶的多项式得到模型并不合理。

样条曲线是一种非参数回归的曲线拟合方法。光滑样条为分段的三次多项式, 曲线在每一段内是一个三次多项式, 在两段的连接点是连续、光滑的。为拟合样条曲线, 只要选"Curves - Spline", 使用缺省的GCV准则(广义交叉核实)来选取光滑系数(光滑系数c越大, 得到的曲线越光滑, 但拟合同时变差, 光滑系数c小的时候得到的曲线较曲折, 而拟合较好), 就可以在散点图的基础上画出样条曲线。可以用光滑系数c的滑块来调整曲线的光滑程度/拟合优度。对于本例, GCV准则得到的样条曲线与回归直线几乎是重合的, 说明直线拟合可以得到满意的结果。

核估计是另一种非参数回归的曲线拟合方法。它定义了一个核函数$K(x)$, 例如使用标准正态分布密度作核$K(x)$, 然后用如下公式估计经验公式$f(x)$:

$$\hat{f}(x) = \sum_{i=1}^{N} K\left(\frac{x - X_i}{\lambda}\right) Y_i$$

其中λ为窗宽, λ越大得到的曲线越光滑。为了画核估计曲线, 只要选"Curves - Kernel", 核函数使用缺省的正态核, 选取光滑系数的方法采用缺省的GCV法, 就可以把核估计图附加到散点图上。本

例得到的核估计曲线与回归直线、样条曲线有一定差别。可以手动调整光滑系数c的值, 可以看到, 当c过大时曲线不仅变光滑而且越来越变水平, 因为这时的拟合值基本是一个常数, 这与样条曲线的情形不同, 样条曲线当c增大时曲线变光滑但不趋向于常数(水平线)。

局部多项式估计(Loess)是另一种非参数回归的曲线拟合方法。它在每一自变量值处拟合一个局部多项式, 可以是零阶、一阶、二阶, 零阶时与核估计相同。SAS/INSIGHT缺省使用一阶(线性)局部多项式。改变Loess的系数alpha可以改变曲线的光滑度。alpha增大时曲线变光滑, 而且使用一阶或二阶多项式时曲线不会因为加大alpha而变水平。

固定带宽的局部多项式是另一种局部多项式拟合方法。它有一个光滑系数c。

4.2.2 用SAS/INSIGHT进行线性回归分析

上面我们已经看到, 用菜单“Analyze - Fit (Y X)”就可以拟合一条回归直线, 这是对回归方程

$$y = a + bx + \varepsilon$$

的估计结果。这样的线性回归可以推广到一个因变量、多个自变量的情况。线性模型写成矩阵形式为

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon$$

其中$\mathbf{Y}$为$n \times 1$向量, $\mathbf{X}$为$n \times p$矩阵, 一般第一列元素全是1, 代表截距项。β为$p \times 1$未知参数向量, ε为$n \times 1$随机误差向量, ε的元素独立且方差为相等的σ^2(未

知)。正常情况下, 系数的估计为$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$, 拟合值(或称预报值)为$\hat{\mathbf{Y}} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} = \mathbf{H}\mathbf{Y}$, 其中$\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ 是R^n 空间的向量向$\mathbf{X}$ 的列张成的线性空间$\mu(\mathbf{X})$投影的投影算子矩阵, 叫做"帽子"矩阵。拟合残差为$\hat{\varepsilon} = \mathbf{Y} - \hat{\mathbf{Y}} = (\mathbf{I} - \mathbf{H})\mathbf{Y}$, 残差平方和为$\text{ESS} = \hat{\varepsilon}'\hat{\varepsilon} = \sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$, 误差项方差的估计(要求设计阵$\mathbf{X}$满秩)为均方误差(MSE) $s^2 = \text{MSE} = \frac{1}{n-p}\text{ESS}$, 在线性模型的假设下, 若设计阵$\mathbf{X}$满秩, $\hat{\beta}$和s^2 分别是β和σ^2的无偏估计, 系数估计的方差阵$\text{Var}(\hat{\beta}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-1}$。判断回归结果优劣的一个重要指标为复相关系数平方(决定系数) $R^2 = 1 - \frac{\text{ESS}}{\text{TSS}}$ (其中$\text{TSS} = \sum_{i=1}^{n}\left(Y_i - \bar{Y}\right)^2$), 它代表在因变量的变差中用模型能够解释的部分的比例, 所以R^2越大说明模型越好。

例如, 我们在"Fit (Y X)"的选择变量窗口选Y变量(因变量)为体重(WEIGHT), 选X变量(自变量)为身高(HEIGHT)和年龄(AGE), 则可以得到体重对身高、年龄的线性回归结果。下面对基本结果进行说明。

回归基本模型:

```
weight = height age
Response Distribution: Normal
Link Function: Identity
```

回归模型方程:

```
Model Equation
weight = -141.2238 + 3.5970 height + 1.2784 age
```

拟合概况：

Summary of Fit			
Mean of Response	100.0263	R-Square	0.7729
Root MSE	11.5111	Adj R-Sq	0.7445

其中Mean of Response为因变量(Response)的均值, Root MSE叫做根均方误差, 是均方误差的平方根, R-Square即复相关系数平方, Adj R-Sq为修正的复相关系数平方, 其公式为$\tilde{R}^2 = 1 - \frac{n-i}{n-p}(1-R^2)$, 其中$i$当有截距项时取1, 否则取0, 这个公式考虑到了自变量个数p的多少对拟合的影响, 原来的R^2随着自变量个数的增加总会增大, 而修正的$\tilde{R}^2$则因为p对它有一个单调减的影响所以p增大时修正的$\tilde{R}^2$不一定增大, 便于不同自变量个数的模型的比较。

方差分析表：

Analysis of Variance					
Source	DF	Sum of Squares	Mean Square	F Stat	Pr > F
Model	2	7215.6371	3607.8186	27.23	<.0001
Error	16	2120.0997	132.5062		
C Total	18	9335.7368			

这是关于模型是否成立的最重要的检验。它检验的是H_0：模型中所有斜率项系数都等于零, 这等价于说自变量的线性组合对因变量没有解释作用。它依据的是一个标准的方差分解, 把因变量的总离差平方和(C Total)分解为能用模型解释的部分(Model)与不能被模型解释的部分(随机误差, Error)之和, 如果能解释的部分占的比例大就否定H_0。F统计量(F Stat)就是这个比例(用自由度修正过)。从上面结果看我们这个模型很显著(p值小于万分之一), 所以可以否定H_0, 模型是有意义的。

第三类检验：

Type III Tests					
Source	DF	Sum of Squares	Mean Square	F Stat	Pr > F
height	1	2091.1460	2091.1460	15.78	0.0011
age	1	22.3880	22.3880	0.17	0.6865

这个表格给出了对各个斜率项是否为零($H_0 : \beta_j = 0$)的检验结果。检验利用的是所谓第三类平方和(Type III SS), 又叫偏平方和, 它代表在只缺少了本变量的模型中加入本变量导致的模型平方和的增加量。比如, HEIGHT的第三类平方和即现在的模型平方与不包含变量HEIGHT的模型计算的模型平方和之差。第三类平方和与模型中自变量的次序无关, 一般也不构成模型平方和的平方和分解。表中用F统计量对假设进行了检验, 分子是第三类平方和的均方, 分母为误差的均方。实际上, 当分子自由度为1时, F统计量即通常的t检验统计量的平方。从表中可见, 身高的作用是显著的, 而年龄的作用则不显著, 有可能去掉年龄后的模型更好一些。

参数估计及相关统计量：

Parameter Estimates							
Variable	DF	Estimate	Std Error	t Stat	Pr>|t|	Tolerance	Var Inflation
Intercept	1	-141.2238	33.3831	-4.23	0.0006	.	0.0000
height	1	3.5970	0.9055	3.97	0.0011	0.3416	2.9276

对截距项系数和各斜率项系数, 给出了自由度(DF), 估计值(Estimate), 估计的标准误差(Std Error), 检验系数为零的t统计量, t统计量的p值, 检验共线性的容许度(Tolerance)和方差膨胀因子(Var Inflation)。其中自变量X_i的容许度定义为1减去X_i对其它自变量的复相关系数平方, 因此容许度越小(接近0), 说明X_i对

其它自变量的复相关系数平方越大, 即X_i可以很好地被其它自变量的线性组合近似, 这样X_i在模型中的作用不大。记$\mathbf{C} = (c_{ij})_{n\times n} = (\mathbf{X}'\mathbf{X})^{-1}$, 则$\mathrm{Var}(\hat{\beta}_i) = \sigma^2 c_{ii}$, c_{ii}叫做方差膨胀因子, 它代表X_i的系数估计的方差的比例系数, 显然其值越大说明估计越不准确, 也说明X_i在模型中的作用不大。方差膨胀因子与容许度互为倒数。

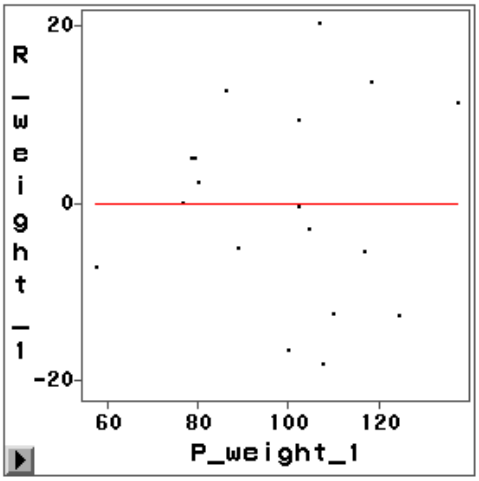


图 4.4: 残差对预测值的散点图

下一个结果为残差对预测值的散点图, 用它可以检验残差中有无异常情况, 比如非线性关系、异方差、模型辨识错误、异常值、序列相关等等。此例中各散点较随机地散布在0线的上下, 没有明显的模式, 可认为结果是合适的(可舍弃的不显著的变量AGE并不反映在残差图中)。

图4.5–4.8 是典型的非线性、异方差、异常值、序列相关的情况的残差图。

用Tables菜单可以加入一些其它的统计量, 如做共线诊断(Colinearity Diagnostics)的条件数(Conditional

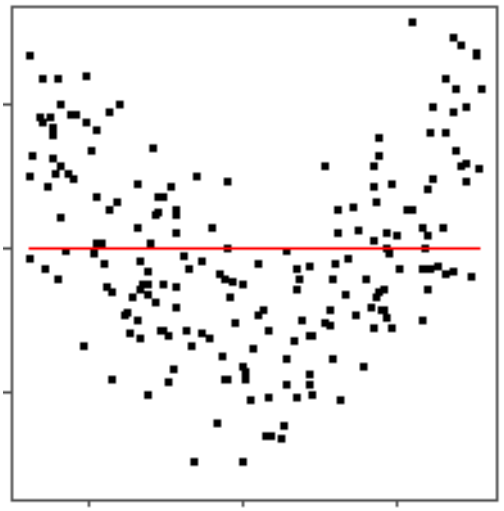


图 4.5: 非线性

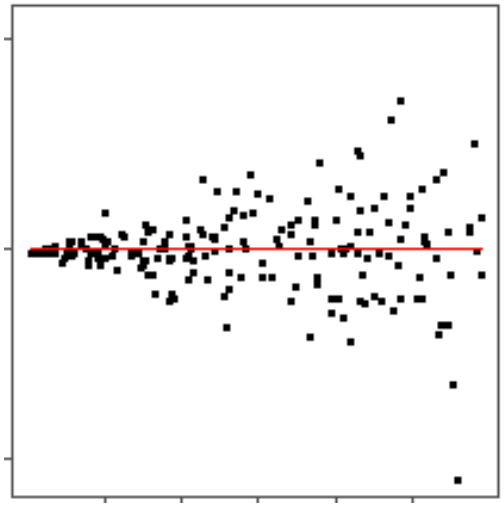


图 4.6: 异方差

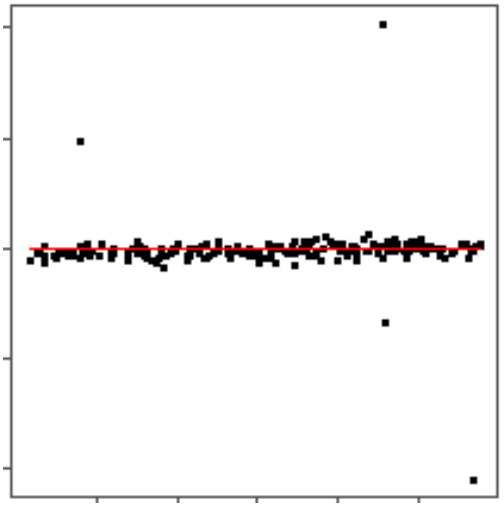


图 4.7: 异常值

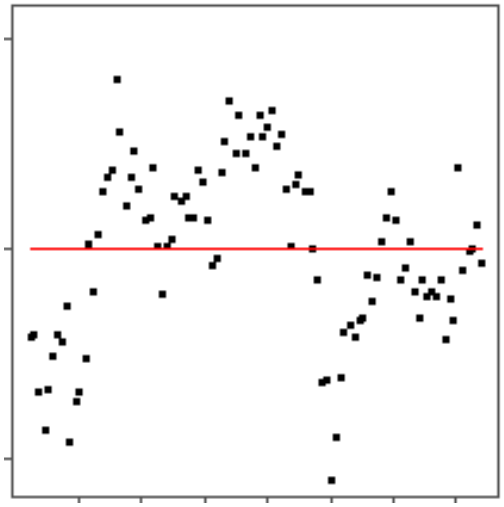


图 4.8: 序列相关

Index)。用Graphs菜单可以加入残差的正态概率图(Residual Normal QQ)和偏杠杆图(Partial Leverage)。

在Vars菜单中可以指定一些变量, 这些变量可以加入到数据窗口中。数据窗口的内容保存在内存中, 不自动改写磁盘中的数据集, 所以要保存数据窗口的修改结果的话需要用"File - Save - Data"命令指定一个用来保存的数据集名。为了了解加入的变量的具体意义, 选数据窗口菜单中的"Data Options", 选中"Show Variable Labels"选项。各变量中, **Hat Diag**为帽子矩阵的对角线元素(帽子矩阵**H**恰好是$n \times n$的), 即杠杆率, 反映了每个观测的影响大小。**Predicted**为拟合值(预报值), **Linear Predictor**为使用线性模型拟合的结果, 在线性回归时与Predicted相同。**Residual**为残差。**Residual Normal Quantile**是残差由小到大排序后对应的标准正态的分位数, 第i个残差的正态分位数用$\Phi^{-1}\left(\frac{i-0.375}{n+0.25}\right)$计算, 其中$\Phi$为标准正态分布函数, 参见1.3.7关于QQ图的解释。**Standardized Residual**(标准化残差)为残差除以其标准误差。**Studentized Residual**(学生化残差)为与标准化残差类似, 但计算第i个学生化残差时预测值和方差估计都是在删除第i个观测后得到的。当学生化残差的值超过2时这个观测有可能是强影响点或异常点。

关于其它的一些诊断统计量请参考帮助菜单的"SAS System Help - Help on SAS Software Products - SAS/INSIGHT Software - Multiple Regression",

或《SAS系统：SAS/STAT软件使用手册》第一章和第九章。

在SAS/INSIGHT中, 为了保存结果表格, 在进行分析之前选中菜单“File - Save - Initial Tables”, 这是一个状态开关, 选中时输出表格画在分析窗口内的同时显示在输出(Output)窗口。如果要保存某一个表格, 也可以选定此表格(单击表格外框线), 然后用菜单“File - Save - Tables”。为了保存分析窗口的图形, 先选定此图形, 然后选“File - Save - Graphics File”, 输入一个文件名, 选择一种文件类型如BMP即可。为了打印某一表格或图形, 先选定它, 然后用菜单“File - Print”。选中“File - Save - Statements”可以开始保存SAS/INSIGHT程序语句。

4.2.3 用SAS/INSIGHT拟合广义线性模型

经典线性回归理论的估计与假设检验要求自变量$\mathbf{X}$非随机, 随机误差项满足$\varepsilon \sim N(0, \sigma^2 I)$。广义线性模型放宽了这些假设, 其模型为

$$\begin{aligned} \mathbf{Y} &= \mu + \varepsilon \\ \eta &= g(\mu) = \eta_0 + \mathbf{X}\beta \end{aligned}$$

其中因变量$\mathbf{Y}$($n \times 1$向量)的元素为服从指数族分布(如正态、逆高斯、伽马、泊松、二项分布)的随机变量, 随机误差项ε($n \times 1$向量)的元素与$\mathbf{Y}$的元素分布类型相同, 元素之间相互独立, 单调函数$g(\cdot)$叫做联系函数(Link function), 它把因变量的均值μ与自变量$\mathbf{X}$ ($n \times p$阵)列的线性组合联系起来。β($p \times$

1向量)为回归系数。模型中每个自变量对应于设计阵$\mathbf{X}$中的一列或几列, $\mathbf{X}$的第一列一般元素全为1, 对应于截距项。$\eta_0(n \times 1$向量)是表示偏移量的变量。

注: 随机变量Y称为服从指数族分布, 如果其分布密度(概率函数)有如下形式:

$$f(y;\theta,\phi) = \exp\left(\frac{\theta y - b(\theta)}{a(\phi)} + c(y,\phi)\right)$$

其中θ为自然参数或称经典参数, ϕ为分散度参数(与尺度参数有关), a, b, c为确定性函数。这样的随机变量Y的均值和方差与参数的关系如下:

$$\begin{aligned} \mathrm{E}(Y) &= b'(\theta) \\ \mathrm{Var}(Y) &= a(\phi)b''(\theta) \end{aligned}$$

为了使用SAS/INSIGHT拟合广义线性模型, 在选"Analyze - Fit (Y X)"之后, 在拟合的对话框中先选定因变量和自变量, 然后按"Method"按钮, 出现选择模型的对话框(图4.9), 在这里可以选因变量的分布类型(Response Dist.), 选联系函数, 选估计尺度参数的方法。

各联系函数定义如下:

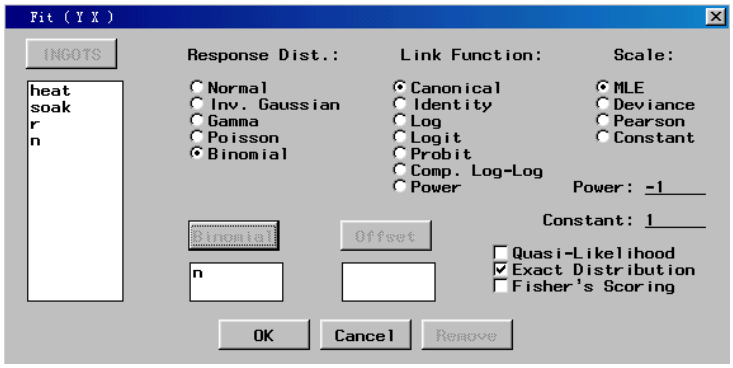


图 4.9: 广义线性模型

Identity	恒等变换
Log	自然对数
Logit	$g(\mu)=\log\frac{\mu}{1-\mu}, 0<\mu<1$
Probit	$g(\mu)=\Phi^{-1}(\mu)$, 其中Φ为标准正态分布函数
Comp. Log-Log	重对数变换$g(\mu)=\log(-\log(1-\mu)), 0<\mu<1$
Power	$g(\mu)=\begin{cases}\mu^{\lambda} & \lambda\neq 0\\ \log(\mu) & \lambda=0\end{cases}$, λ在对话框的Power输入框指定。

指数族中每一个分布有一个特定的联系函数, 使得$\eta=g(\mu)=\theta$, 即用分布的期望值表示经典参数, 这样的联系函数叫经典(canonical)联系函数。正态分布的经典联系函数为恒等变换, 逆高斯分布为-2次方变换, 伽玛分布为-1次方变换, 泊松分布为对数变换, 二项分布为逻辑变换(Logit)。注意Logit、probit、重对数变换都只适用于二项分布。

例如, SASUSER.INGOTS中存放了一个铸造厂的数据, 它记录了各批铸件在一定的加热、浸泡时间

条件下出现的不能开始轧制的铸件数目。HEAT为加热时间, SOAK为浸泡时间, N为每批铸件的件数, R为加热浸泡后N件铸件中还不能开始轧制的铸件数。R应该服从二项分布, 其分布参数(比例)可能受加热、浸泡时间的影响。因此, 我们拟合以R为因变量, 以HEAT和SOAK为自变量的广义线性模型, 因变量分布为二项分布, 使用经典联系函数(Logit函数)。模型为

$$\begin{aligned} R &\sim B(N,\mu) \\ \log\frac{\mu}{1-\mu} &= \beta_0+\beta 1\cdot \mathrm{HEAT}+\beta_2\cdot \mathrm{SOAK} \end{aligned}$$

为了拟合这样的模型, 选“Analyze - Fit(Y X)”, 选R为因变量, 选HEAT和SOAK为自变量, 按“Method”钮, 选因变量分布为二项分布(Binomial), 选变量N然后按“Binomial”钮, 两次OK后即可以得到模型拟合窗口。可以看到, 这个模型是显著的, 但变量SOAK没有显著影响。去掉变量SOAK重新拟合模型。HEAT的系数为0.0807是正数, 说明加热时间越长不能轧制的件数越多。考察拟合结果窗口下方的残差对预报值图可以发现在右下方有三个异常点, 用刷亮方法选定它们, 可以看到, 这三个观测都是总共只有一个铸件的, 所以对一般结果意义不大。选“Edit - Observations - Exclude in Calculation”可以把这几个点排除在外, 发现结果基本不变。

4.2.4 用REG过程进行回归分析

SAS/STAT中提供了几个回归分析过程, 包括REG(回归)、RSREG(二次响应面回归)、ORTHOREG(病态数据回归)、NLIN(非线性回归)、TRANSREG(变换回归)、CALIS(线性结构方程和路径分析)、GLM(一般线性模型)、GENMOD(广义线性模型), 等等。我们这里只介绍REG过程, 其它过程的使用请参考《SAS系统——SAS/STAT软件使用手册》。

REG过程的基本用法为：

PROC REG DATA=输入数据集 *选项*;
VAR *可参与建模的变量列表*;
MODEL 因变量＝自变量表/ *选项*;
PRINT 输出结果;
PLOT 诊断图形;
RUN;

REG过程是交互式过程, 在使用了RUN语句提交了若干个过程步语句后可以继续写其它的REG过程步语句, 提交运行, 直到提交QUIT语句或开始其它过程步或数据步才终止。

例如, 我们对SASUSER.CLASS中的WEIGHT用HEIGHT和AGE建模, 可以用如下的简单REG过程调用：

```
proc reg data=sasuser.class;
  var weight height age;
  model weight=height age;
run;
```

就可以在输出窗口产生如下结果, 注意程序窗口的标题行显示“PROC REG Running”表示REG过程还在运行, 并没有终止。

```
                    The REG Procedure
                      Model: MODEL1
          Dependent Variable: weight Weight in pounds

                    Analysis of Variance

                         Sum of          Mean
Source            DF      Squares       Square  F Value    Pr > F

Model              2   7215.63710   3607.81855    27.23    <.0001
Error             16   2120.09974    132.50623
Corrected Total   18   9335.73684


            Root MSE             11.51114    R-Square     0.7729
            Dependent Mean      100.02632    Adj R-Sq     0.7445
            Coeff Var            11.50811


                          Parameter Estimates

                               Parameter Standard
 Variable   Label            DF   Estimate     Error t Value  Pr > |t|

 Intercept  Intercept         1 -141.22376  33.38309   -4.23    0.0006
 height     Height in inches  1    3.59703   0.90546    3.97    0.0011
 age        Age in years      1    1.27839   3.11010    0.41    0.6865
```

这些结果与SAS/INSIGHT得到的结果是一致的。同样我们发现变量AGE的作用不显著, 所以我们只要再提交如下语句：

```
model weight=height;
run;
```

就可以得到第二个模型结果：

```
                Model: MODEL2
Dependent Variable: weight Weight in pounds
.....................
```

事实上, REG提供了自动选择最优自变量子集的选项。在MODEL语句中加上"SELECTION=*选择方法*"的选项就可以自动挑选自变量, 选择方法有NONE(全用, 这是缺省), FORWARD(逐步引入法), BACKWARD(逐步剔除法), STEPWISE(逐步筛选法), MAXR(最大R^2增量法), MINR(最小R^2增量法), RSQUARE(R^2选择法), ADJRSQ(修正R^2选择法), CP(Mallows的C_p统计量法)。比如, 我们用如下程序:

```
model weight=height age / selection=stepwise;
run;
```

可得到如下结果:

```
                          The SAS System

                        The REG Procedure
                          Model: MODEL1
            Dependent Variable: weight Weight in pounds

                    Stepwise Selection: Step 1

      Variable height Entered: R-Square = 0.7705 and C(p) = 1.1690


                     Analysis of Variance

                        Sum of         Mean
 Source           DF     Squares       Square  F Value   Pr > F

 Model             1  7193.24912   7193.24912    57.08   <.0001
 Error            17  2142.48772    126.02869
 Corrected Total  18  9335.73684


              Parameter     Standard
 Variable      Estimate        Error   Type II SS  F Value  Pr > F

 Intercept   -143.02692     32.27459   2475.04718    19.64  0.0004
 height         3.89903      0.51609   7193.24912    57.08  <.0001

               Bounds on condition number: 1, 1
------------------------------------------------------------------


   All variables left in the model are significant at the 0.1500 level.
```

```
No other variable met the 0.1500 significance level for entry into the model.


                    Summary of Stepwise Selection

     Variable  Variable                  Number  Partial   Model
Step Entered   Removed   Label            Vars In R-Square R-Square  C(p)

  1  height              Height in inches     1    0.7705   0.7705   1.1690

                  Summary of Stepwise Selection

                     Step F Value    Pr > F

                        1    57.08    <.0001
```

可见只有变量HEIGHT进入了模型, 而其它变量(AGE)则不能进入模型。

REG过程给出的缺省结果比较少。用PRINT语句和PLOT语句可以显示额外的结果。为了显示模型的预测值(拟合值)和95%预测界限, 使用语句

```
print cli;
run;
```

得到如下的结果:

```
                          The REG Procedure
                            Model: MODEL1
            Dependent Variable: weight Weight in pounds

                          Output Statistics

       Dep Var  Predicted     Std Error
Obs     weight      Value  Mean Predict     95% CL Predict      Residual

  1    84.0000    77.2683        3.9633    52.1503   102.3863     6.7317
  2    98.0000   111.5798        2.9953    87.0659   136.0936   -13.5798
  3    90.0000   107.6807        2.7676    83.2863   132.0752   -17.6807
  4    77.0000    76.4885        4.0423    51.3145   101.6624     0.5115
  5    84.5000    90.1351        2.8892    65.6780   114.5922    -5.6351
  6   112.0000   116.2586        3.3540    91.5388   140.9784    -4.2586
  7    50.5000    56.9933        6.2512    29.8835    84.1032    -6.4933
  8   112.5000   100.6625        2.5769    76.3612   124.9637    11.8375
  9   102.5000   101.8322        2.5865    77.5263   126.1380     0.6678
 10   112.5000   126.0062        4.2963   100.6456   151.3668   -13.5062
 11   102.5000   104.5615        2.6445    80.2279   128.8951    -2.0615
 12   133.0000   118.2081        3.5249    93.3827   143.0335    14.7919
```

```
 13    83.0000    80.3875        3.6593    55.4757   105.2993     2.6125
 14    84.0000   100.6625        2.5769    76.3612   124.9637   -16.6625
 15    99.5000    87.0159        3.0982    62.4451   111.5866    12.4841
 16   150.0000   137.7033        5.6129   111.2225   164.1840    12.2967
 17   128.0000   109.6302        2.8721    85.1821   134.0784    18.3698
 18    85.0000    81.1673        3.5867    56.3025   106.0321     3.8327
 19   112.0000   116.2586        3.3540    91.5388   140.9784    -4.2586


           Sum of Residuals                           0
           Sum of Squared Residuals          2142.48772
           Predicted Residual SS (PRESS)     2651.35206
```

各列分别为观测序号(Obs), 因变量的值(Dep Var), 预测值(Predicted Value), 预测值的期望值的标准误差(Std Error Mean Predict), 预测值的95%置信区间(95% CL Predict), 残差(Residual, 为因变量值减预测值)。在表后又给出了残差的总和(Sum of Residuals), 残差平方和(Sum of Squared Residuals), 预测残差的平方和(Predicted Resid SS (Press))。所谓预测残差, 是在计算第i号观测的残差时从实际值中减去的预报值是用扣除第i号观测后的样本得到的模型产生的预报值, 而不是我们一般所用的预测值(实际是拟合值)。第i号样本的预测残差还可以用公式$\text{PRESSID}_i = \text{RESID}_i/(1-h_i)$来计算, 其中$\text{RESID}_i$是第$i$个残差，$h_i$为帽子矩阵$\mathbf{H}$的第$i$个主对角线元素。

用print cli列出的是预测值的置信区间, 还可以列出模型均值的置信区间, 使用

```
print clm;
run;
```

语句。在PRINT语句中可以指定的有ACOV, ALL, CLI, CLM, COLLIN, COLLINOINT, COOKD, CORRB, COVB, DW, I, INFLUENCE, P, PARTIAL,

PCORR1, PCORR2, R, SCORR1, SCORR2, SEQB, SPEC, SS1, SS2, STB, TOL, VIF, XPX, 等等。

对于自变量是一元的情况, 可以在自变量和因变量的散点图上附加回归直线和均值置信界限。比如,

```
plot weight * height / conf95;
run;
```

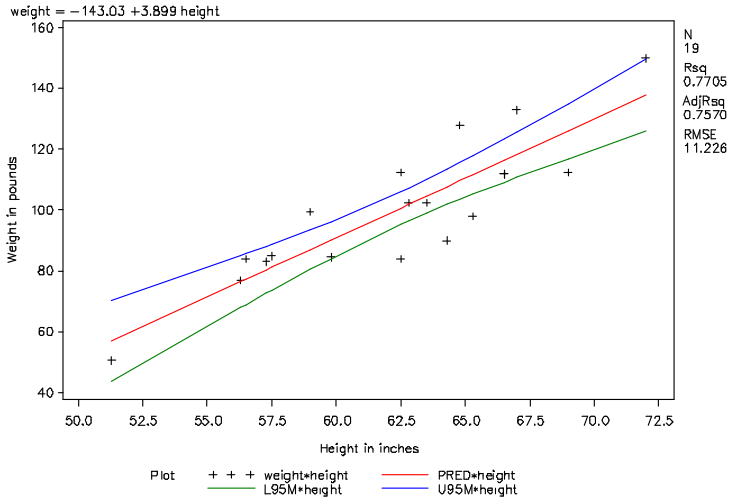


图 4.10: REG过程绘图

可以产生4.10, 在图的上方列出了模型方程, 右方还给出了观测个数、R^2、修正的R^2、根均方误差。在PLOT语句中可以使用“PREDICTED.”、“RESIDUAL.”等特殊名字表示预测值、残差等计算出的变量, 比如, 在自变量为多元时无法作回归直线, 常用的诊断图表为残差对预测值图, 就可以用

```
plot residual. * predicted.;
run;
```

绘制。为了绘制学生化残差的图形, 可以用

```
plot rstudent. * obs.;
run;
```

回归分析的其它用法及进一步的诊断方法请参考有关统计书籍和SAS使用手册。

4.2.5 用Analyst进行回归分析

我们可以用Analyst 的图形界面调用回归分析功能。“Statistics - Regression”菜单提供了三种回归：一元回归、线性回归和Logistic回归。一元回归可以是通常的一元线性回归，也可以拟合二次或三次多项式。线性回归可以完成REG过程的大部分功能。

我们以SASUSER.CLASS数据集为例。选了“Statistics - Regression - Linear”之后出现如图4.11的对话框，要指定因变量和自变量。这个对话框的Model按钮可以指定模型选择方法及具体选择方法的细节。Statisics按钮可以要求输出与模型拟合优度和模型诊断有关的统计量。Predictions可以要求计算对数据集中各观测的预测值、残差值、预测界限，也可以指定一个包含模型自变量的数据集要求对其进行预测。Plots可以要求画各种回归诊断图形，如残差图、杠杆图等。Save Data可以把指定的结果保存到数据集中。

4.3 方差分析入门

统计学中用方差分析来研究分类变量(所谓“因

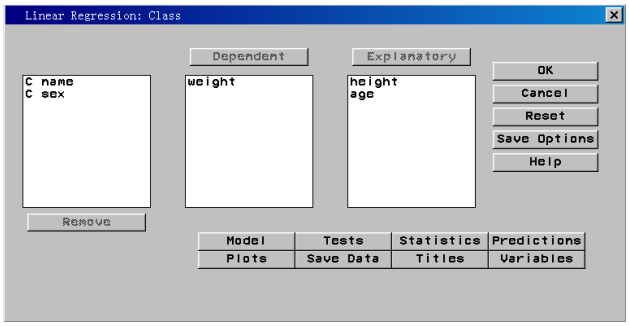


图 4.11: Analyst: 线性回归

素")对数值型变量(所谓"指标")的影响。主要目的是研究某些因素对于指标有无显著的影响。对有显著影响的因素, 一般希望找出最好水平。

4.3.1 用ANOVA过程进行单因素方差分析

单因素方差分析是4.1.2问题的一个自然延续。在4.1.2中, 我们有一个分类变量把观测分为两组, 我们要研究这两组的均值有没有显著差异。如果这个分类变量的取值不只两个, 则这时4.1.2的检验方法不再适用, 但我们同样要解决各组均值是否有显著差异的问题。如果各组之间有显著差异, 说明这个因素(分类变量)对指标是有显著影响的, 因素的不同取值(叫做水平)会影响到指标的取值。

例如, 数据集SASUSER.VENEER是关于若干种牌子的胶合板的耐磨情况数据, 变量BRAND为试样的牌子, 变量WEAR为试样的磨损量。共有五种牌子的胶合板, 每种试验了4个试样。我们希望知道这五种牌子胶合板的磨损量有无显著差别, 如果无显著差别我们在选购时就不必考虑哪一个更耐磨而只需考虑价格等因素, 但如果结果有显著差异则应考虑使用耐

磨性好的牌子。这里, 因素是胶合板的牌子, 指标为磨损量, 当各种牌子胶合板磨损量有显著差异时, 说明因素的取值对指标有显著的影响。所以, 方差分析的结论是因素对指标有无显著影响。注意, 经典的方差分析只判断因素的各水平有无显著差异, 而不管两个水平之间是否有差异, 比如说我们的五个牌子即使有四个牌子没有显著差异, 只有一个牌子的胶合板比这四个牌子的都好, 结论也是说因素是显著的, 或因素的各水平间有显著差异。

方差分析把指标的方差分解为由因素的不同取值能够解释的部分, 和剩余的不能解释的部分, 然后比较两部分, 当能用因素解释的部分明显大于剩余的部分时认为因素是显著的。方差分析假定观测是彼此独立的, 观测为正态分布的样本, 由因素各水平分成的各组的方差相等。在这些假定满足时, 就可以用ANOVA过程来进行方差分析。其一般写法为

```
PROC ANOVA DATA＝数据集;
        CLASS 因素;
        MODEL 指标＝因素;
RUN;
```

比如, 为了分析SASUSER.VENEER中各种牌子的胶合板的耐磨性有无显著差别, 首先我们假定假设检验使用的检验水平为0.05, 可以使用如下程序进行方差分析：

```
proc anova data=sasuser.veneer;
  class brand;
  model wear=brand;
run;
```

结果如下：

```
                    The ANOVA Procedure

                  Class Level Information

      Class         Levels    Values

      brand              5    ACME AJAX CHAMP TUFFY XTRA

                Number of observations    20


Dependent Variable: wear   Amount of material worn away

                                      Sum of
Source             DF     Squares  Mean Square  F Value  Pr > F

Model               4  0.61700000   0.15425000     7.40  0.0017

Error              15  0.31250000   0.02083333

Corrected Total    19  0.92950000


          R-Square     Coeff Var      Root MSE     wear Mean

          0.663798      6.155120      0.144338      2.345000


Source             DF    Anova SS  Mean Square  F Value  Pr > F

brand               4  0.61700000   0.15425000     7.40  0.0017
```

结果可以分为四个部分, 第一部分是因素水平的信息, 我们看到因素只有一个BRAND, 它有5个水平, 分别 是ACME、AJAX、CHAMP、TUFFY、XTRA。共有20个观测。第二部分就是经典的方差分析表, 表前面指明了因变量(指标)为WEAR, 第一列“来源(Source)”说明方差的来源, 是模型(Model)的(可以用方差分析模型解释的), 误差(Error)的(不能用模型解释的), 还是总和(Corrected Total)。第三列为平方

和, 其大小代表了各方差来源作用的大小。第二列为自由度。第四列为均方, 即平方和除以自由度。第五列F值是F统计量的值, 其计算公式为模型均方除以误差均方, 用来检验模型的显著性, 如果不显著说明模型对指标的变化没有解释能力。第六列是F统计量的p值。由于这里p值小于0.05(我们的检验水平), 所以模型是显著的, 因素对指标有显著影响。结果的第三部分是一些与模型有关的简单统计量, 第一个是复相关系数平方, 与回归模型一样仍代表总变差中能被模型解释的比例, 第二个是变异系数, 第三个是根均方误差, 第四个是指标的均值。结果的第四部分是方差分析表的细化, 给出了各因素的平方和和F统计量, 因为是单因素所以这一行与上面的“模型”一行相同。

4.3.2 用NPAR1WAY进行非参数单因素方差分析

当方差分析的正态分布假定或方差相等假定不能满足时, 对单因素问题, 可以使用称为Kruskal-Wallis检验的非参数方差分析方法。这种检验不要求观测来自正态分布总体, 不要求各组的方差相等, 甚至指标可以是有序变量(变量取值只有大小之分而没有差距的概念, 比如磨损量可以分为大、中、小三档, 得病的程度可以分为重、轻、无, 等等)。

NPAR1WAY过程的调用与ANOVA过程不同, 因为它是单因素方差分析过程, 所以只要用CLASS语句给出分类变量(因素), 用VAR语句给出指标就可以了, 一般格式为：

```
PROC NPAR1WAY DATA=数据集 WILCOXON;
    CLASS 因素;
    VAR 指标;
RUN;
```

注意这样的语句格式与4.1.2中两独立样本比较的做法完全相同。NPAR1WAY过程当“因素”有两个水平时, 执行Wilcoxon秩和检验, 多个水平时执行Kruskal-Wallis检验。

比如, 为了分析上面的胶合板例子中各牌子的耐磨性有无显著差异, 取定0.10的检验水平, 可以用如下的NPAR1WAY过程：

```
proc npar1way data=sasuser.veneer wilcoxon;
  class brand;
  var wear;
run;
```

得到如下结果：

```
                The NPAR1WAY Procedure

     Wilcoxon Scores (Rank Sums) for Variable wear
            Classified by Variable brand

              Sum of  Expected     Std Dev     Mean
  brand   N   Scores  Under H0    Under H0    Score
  ------------------------------------------------
  ACME    4     40.0      42.0   10.483069   10.000
  CHAMP   4     44.0      42.0   10.483069   11.000
  AJAX    4     12.0      42.0   10.483069    3.000
  TUFFY   4     69.0      42.0   10.483069   17.250
  XTRA    4     45.0      42.0   10.483069   11.250

          Average scores were used for ties.


                  Kruskal-Wallis Test
```

```
Chi-Square          11.9824
DF                        4
Pr > Chi-Square      0.0175
```

结果分为两个部分, 第一部分是各组的秩和的情况, 包括观测个数(N)、秩和(Sum of Scores)、在各组无显著差异的零假设下的期望秩和(Expected Under H0)、在零假设下的标准差(Std Dev Under H0)、平均秩(Mean Score, 为秩和除以组内观测数)。所谓秩(Rank)就是从小到大排列的名次。下面的"Average scores were used for ties"是说当名次相同时(如两个第2名)用名次的平均值((2+3)/2=2.5)。第二部分为Kruskal-Wallis检验的结果, 包括近似的χ^2统计量, 自由度, 检验的p值(Prob > Chi-Square)。现在p值0.0175小于预定的水平0.10所以结论是各种牌子的胶合板的耐磨性能有显著差异。注意, Kruskal-Wallis检验是非参数检验, 在同等条件下非参数检验一般比参数检验的功效低, 所以这里的p值0.0175比用ANOVA过程得到的p值0.0017要大。

4.3.3 多重比较

方差分析只检验各组是否没有任何两两之间的差异, 但不检验到底是哪两组之间有显著差异。在三个或多个组之间进行两个或多个比较的检验叫做多重比较。多重比较在统计学中没有一个公认的解决方法, 而是提供了若干种检验方法。因为多重比较要进行不只一次的比较, 所以在多重比较的检验水平有两种：总错误率(experimentwise error rate)和单次比较错误率(comparisonwise error rate)。总错误率是指所

有比较(比如, 五个组两两之间比较有10次)的总第一类错误概率, 单次比较错误率是指每一次比较的第一类错误概率。显然, 总错误率要比单次比较错误率高。

在ANOVA过程中使用MEANS语句可以进行多重比较。格式如下：

MEANS 因素/ 选项;

如果不使用选项, 则ANOVA过程内的MEANS语句只对因素的各水平计算指标的平均值和标准差, 比如：

```
proc anova data=sasuser.veneer;
  class brand;
  model wear=brand;
  means brand;
run;
```

则在通常的方差分析结果基础上增加如下结果：

```
            The ANOVA Procedure

   Level of          -------------wear------------
   brand       N             Mean          Std Dev

   ACME        4       2.32500000       0.17078251
   AJAX        4       2.05000000       0.12909944
   CHAMP       4       2.37500000       0.17078251
   TUFFY       4       2.60000000       0.14142136
   XTRA        4       2.37500000       0.09574271
```

为了进行两两比较, 可以在MEANS语句的选项中指定检验方法。SAS提供了多种方法。

一、用重复t检验控制单次比较错误率

重复t检验的想法很简单：在适当的检验水平下用两

样本t检验对所有组两两之间进行检验。控制的是每次比较的第一类错误概率。缺省使用0.05水平。注意这样检验的总错误率将大大高于每次比较的错误率。比如, 在上面程序后加入(ANOVA是交互式过程)

```
means brand / t;
run;
```

可得如下结果：

```
            t Tests (LSD) for wear

NOTE: This test controls the Type I comparisonwise
      error rate, not the experimentwise error rate.


        Alpha                            0.05
        Error Degrees of Freedom           15
        Error Mean Square            0.020833
        Critical Value of t           2.13145
        Least Significant Difference   0.2175


Means with the same letter are not significantly different.


  t Grouping          Mean      N    brand

           A        2.6000      4    TUFFY

           B        2.3750      4    XTRA
           B
           B        2.3750      4    CHAMP
           B
           B        2.3250      4    ACME

           C        2.0500      4    AJAX
```

结果先说明了检验的指标是变量WEAR, 然后说明了这种检验控制单次比较的第一类错误概率而不是总的第一类错误概率。下面给出了检验的一些指标, 比

如水平(Alpha)为0.05(控制单次比较的第一类错误概率), 自由度(df)为15, 误差的均方(MSE, 是方差分析表中误差的均方)为0.020833, 两样本t检验的t统计量的临界值(Critical Value of t)为2.13, 如果两样本t检验的t统计量值绝对值超过临界值则认为两组有显著差异, 或者等价地, 如果两组的均值之差绝对值大于最小显著差别(Least Significant Difference)0.2175也是有显著差异。所以这个检验也叫LSD检验。下面列出了检验的结果, 把因素各水平的指标平均值由大到小排列, 然后把两两比较的结果用第一列的字母来表示, 字母相同的水平没有显著差异, 字母不同的水平有显著差异。所以我们看到, 重复t检验的结果把五种牌子分成了A、B、C三个组, TUFFY单独是一组, 它的磨损量最大；XTRA、CHAMP、ACME是一组, 这三种两两之间没有显著差异；AJAX单独是一组, 其磨损量最小。

二、用Bonferroni t检验控制总错误率

Bonferroni t检验通过把每次比较的错误率取得很小来控制总误差率。比如, 共有10次比较时, 把每次比较的错误率控制在0.005就可以保证总错误率不超过0.05, 但是, 这样得到的实际总第一类错误率可能要比预定的水平小得多。在MEANS语句中使用BON语句可以执行Bonferroni t检验, 缺省总错误率控制水平为0.05。对上面的胶合板数据增加如下语句：

```
means brand / bon;
run;
```

结果如下：

```
              Bonferroni (Dunn) t Tests for wear

NOTE: This test controls the Type I experimentwise error rate,
  but it generally has a higher Type II error rate than REGWQ.


           Alpha                              0.05
           Error Degrees of Freedom             15
           Error Mean Square              0.020833
           Critical Value of t             3.28604
           Minimum Significant Difference   0.3354


   Means with the same letter are not significantly different.


      Bon Grouping          Mean      N    brand

                 A        2.6000      4    TUFFY
                 A
            B    A        2.3750      4    XTRA
            B    A
            B    A        2.3750      4    CHAMP
            B    A
            B    A        2.3250      4    ACME
            B
            B             2.0500      4    AJAX
```

结果先说明了检验类型和指标(变量WEAR), 然后说明了检验控制总第一类错误率, 但一般比REGWQ方法的第二类错误概率高(检验功效较低)。下面给出了几个检验用的值。最后给出了Bonferroni t检验的结果, 有相同分组字母的因素水平间无显著差异, 否则有显著差异。我们看到, TUFFY与XTRA、CHAMP、ACME没有显著差异, 与AJAX有显著差异；XTRA、CHAMP、ACME两两之间没有显著差异, 而且与其它两个也都没有显著差异；AJAX与TUFFY有显著差异, 与其它三个

没有显著差异。其分组是有交叉的。最后只发现了TUFFY和AJAX之间有显著差异。

三、用REGWQ检验控制总错误率

用Bonferroni t检验控制总错误率过于保守, 功效较低, 不易发现实际存在的显著差异。REGWQ方法可以控制总错误率并且一般比Bonferroni t检验要好。这种方法执行多阶段的检验, 它对因素水平的各种子集进行检验。在MEANS语句中用REGWQ选项可以进行REGWQ检验。例如, 在前面的例子后再运行

```
means brand/ regwq;
run;
```

结果如下

```
     Ryan-Einot-Gabriel-Welsch Multiple Range Test for wear

 NOTE: This test controls the Type I experimentwise error rate.

               Alpha                          0.05
               Error Degrees of Freedom         15
               Error Mean Square          0.020833

Number of Means           2           3           4           5
Critical Range     0.264828   0.2917322   0.2941581     0.31516

  Means with the same letter are not significantly different.

   REGWQ Grouping          Mean      N    brand

                A        2.6000      4    TUFFY
                A
                A        2.3750      4    XTRA
                A
                A        2.3750      4    CHAMP
                A
                A        2.3250      4    ACME
```

```
B          2.0500       4     AJAX
```

可见它比Bonferroni方法发现了较多的显著差异, 除了TUFFY和AJAX仍有显著差异以外, 还发现XTRA、CHAMP、ACME也都与AJAX有显著差异。

MEANS语句的选项可以同时使用。在MEANS语句中可以用ALPHA=水平值来指定检验的水平。ANOVA过程中还提供了其它的多重比较方法, 请自己参考有关资料。

4.3.4 多因素方差分析

SAS提供了若干个方差分析过程, 可以考虑多个因素、有交互作用、有嵌套等情况的方差分析。用GLM过程还可以用一般线性模型来处理方差分析问题。在这里我们只介绍如何用ANOVA过程进行均衡设计的多因素方差分析。

例如, 为了提高一种橡胶的定强, 考虑三种不同的促进剂(因素A)、四种不同分量的氧化锌(因素B)对定强的影响, 对配方的每种组合重复试验两次, 总共试验了24次, 得到表4.1的结果。

表 4.1: 橡胶配方试验数据

A:促进剂	B:氧化锌			
	1	2	3	4
1	31, 33	34, 36	35, 36	39, 38
2	33, 34	36, 37	37, 39	38, 41
3	35, 37	37, 38	39, 40	42, 44

我们首先把数据输入为SAS数据集。输入的办法可以是直接输入各个观测, 例如：

```
data rubber;
  input A B STREN;
  cards;
1 1 31
1 1 33
1 2 34
1 2 36
1 3 35
1 3 36
1 4 39
1 4 38
2 1 33
..............
;
run;
```

也可以使用如下的直接循环控制的INPUT读取：

```
data rubber;
  do  a=1 to 3;
    do  b=1 to 4;
      do  r=1 to 2;
        input stren @@;
        output;
      end;
    end;
  end;
  cards;
31 33  34 36  35 36  39 38
33 34  36 37  37 39  38 41
35 37  37 38  39 40  42 44
;
run;
```

其中INPUT语句尾部的两个@符号表示多次INPUT语句可以从同一行去读取(否则每

次INPUT语句运行时自动从下一行开始读)。

为了研究两个因素的主效应和交互作用, 使用如下ANOVA过程：

```
proc anova data=rubber;
  class a b;
  model  stren = a b a*b;
run;
```

MODEL语句中中A表示因素A的主效应, B表示因素B的主效应, A*B表示A和B的交互作用。结果如下：

```
                The ANOVA Procedure

            Class Level Information
          Class         Levels    Values
          a                  3    1 2 3
          b                  4    1 2 3 4

          Number of observations    24

Dependent Variable: stren
                                      Sum of
 Source            DF      Squares   Mean Square  F Value  Pr > F
 Model             11  193.4583333    17.5871212    12.06  <.0001
 Error             12   17.5000000     1.4583333
 Corrected Total   23  210.9583333

       R-Square     Coeff Var      Root MSE    stren Mean
       0.917045      3.260152      1.207615      37.04167

 Source           DF      Anova SS   Mean Square  F Value    Pr > F
 a                 2    56.5833333    28.2916667    19.40    0.0002
 b                 3   132.1250000    44.0416667    30.20    <.0001
 a*b               6     4.7500000     0.7916667     0.54    0.7665
```

结果首先给出了因素(Class)的变量名和各水平值, 观测数。然后是总的方差分析表, 指明指标为变量STREN, 给出了模型、误差、总平方和, F统计量

值和p值。从p值可见模型是显著的。为了分析各作用的显著性, 看后面的详细的方差分析表, 它给出了模型中各作用(A、B、A*B)的平方和和检验的F统计量值及p值。可以看出, 两个因素的主效应都是显著的, 交互作用效应不显著。所以, 我们可以重新运行ANOVA过程, 不指定交互作用效应：

```
proc anova data=rubber;
  class a b;
  model stren = a b;
run;
```

这时模型的F统计量变为30.53, 因素A主效应的F统计量变为22.89, 因素B主效应的F统计量变为35.63, 都增大了。两个因素的主效应仍是高度显著的, 说明它们对定强都有显著影响。

为了找到最好的配方, 在前面的ANOVA过程后使用

```
  means a b;
run;
```

可以计算出每种水平下的指标平均值, 因素A(促进剂)在第三水平使指标(定强)最大, 因素B(氧化锌)在第四水平使指标最大, 所以最好的配方是：第三种促进剂, 第四种氧化锌分量。

ANOVA也可以用来分析正交设计的结果。例如, 为了提高某种试剂产品的收率(指标), 考虑如下几个因素对其影响：

A: 反应温度	1 (50℃)	2 (70℃)
B: 反应时间	1 (1小时)	2 (2小时)
C: 硫酸浓度	1 (17%)	2 (27%)
D: 硫酸产地	1 (天津)	2 (上海)
E: 操作方式	1 (搅拌)	2 (不搅拌)

把这五个因素放在$L_8(2^7)$表的五列上, 得到如下的试验方案及结果(见下面的数据步)。用ANOVA过程可以分析：

```
data  exp;
  input temp time conc manu mix  prod;
  cards;
1 1 1 1 1  65
1 1 1 2 2  74
1 2 2 1 2  71
1 2 2 2 1  73
2 1 2 1 2  70
2 1 2 2 1  73
2 2 1 1 1  62
2 2 1 2 2  69
;
run;
proc anova data=exp;
  class temp time conc manu mix;
  model prod = temp--mix;
  means temp--mix / t;
run;
```

用0.05水平, 得到的模型是显著的(模型p值为0.0250), 各因素的检验结果如下：

```
 Source   DF      Anova SS   Mean Square  F Value    Pr > F

 temp      1   10.12500000   10.12500000    16.20    0.0565
 time      1    6.12500000    6.12500000     9.80    0.0887
 conc      1   36.12500000   36.12500000    57.80    0.0169
```

```
manu        1    55.12500000    55.12500000     88.20     0.0111
mix         1    15.12500000    15.12500000     24.20     0.0389
```

可见硫酸浓度、产地、操作方式是显著的, 必须采用它们的最好水平, 温度、时间不显著, 在同等条件下可以优先采用它们的最好水平。从MEANS语句的结果可以知道, 硫酸浓度的最好水平是水平2(27%), 硫酸产地的最好水平是水平2(上海), 操作方式的最好水平是水平2(不搅拌), 反应温度的最好水平是水平1(50°C), 反应时间的最好水平是水平1(1小时)。从以上分析可以得到好的生产方案。

4.3.5 用Analyst作方差分析

Analyst 的"Statistics - ANOVA"菜单提供了七种方差分析方法，我们这里只介绍前三种：单因素方差分析(One-Way ANOVA)、非参数单因素方差分析(Nonparametric One-Way ANOVA)、多因素方差分析(Factorial ANOVA)。

为了对SASUSER.VENEER进行方差分析，选"Statistics - ANOVA - One-Way ANOVA), 弹出如图4.12的对话框，我们要指定因变量(Dependent，即指标)和自变量(Independent, 即因素)。用这里的Tests钮可以选择一些其他的检验，比如对各组方差相等假设的检验，稳健的Welch方差加权方差分析等。Means按钮用来进行多重比较，可以选择多种比较方法，见图4.13。这个菜单调用的是PROC ANOVA，所以最后的结果与上面编程得到的结果一致。

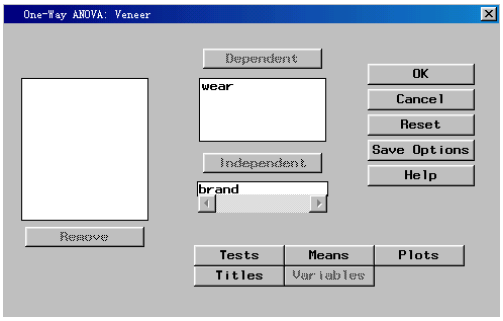


图 4.12: Analyst: 单因素方差分析

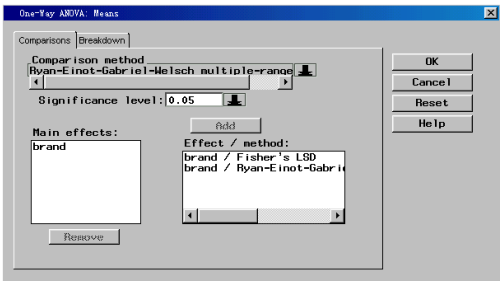


图 4.13: Analyst: 多重比较

用“Statistics - ANOVA - Nonparametric One-Way ANOVA”可以作Kruskal-Wallis检验。它调用PROC NPAR1WAY。

用“Statistics - ANOVA - Factorial ANOVA”可以进行多元方差分析。它调用PROC GLM，这个过程与ANOVA的差别在于它允许非均衡设计。对于均衡设计如上面的橡胶试验这里得到的结果与PROC ANOVA得到的结果是一致的。

4.4 列联表分析

上面所讲到的统计分析主要针对数值型(区间)变量进行。在实际工作中, 离散取值的名义变量(如性别、职业、民族)和有序变量(如调查意见的完全同意、同意、中立、反对、强烈反对) 也是十分常见的, 对这类离散变量(又称属性变量)的分析也是统计学的重要的研究内容。这一节我们讲述检验两个离散取值的变量的独立性的列联表检验方法, 并介绍有序变量的关联性量度。

表 4.2: 学生性别、来源分布表

	男生	女生
本地	4	6
外地	14	7

4.4.1 列联表的输入与制表

离散变量的取值可以把样本进行分类。比如, 我们的样本是一个班的学生情况, 可以根据学生的性别把观测分为男生和女生两个组。我们也可以根据学生的来源把观测分为本地学生和外地学生两个组。如果联合使用这两个变量对观测分类, 就可以把观测分为四个组, 我们可以统计每个组学生的人数, 并把结果画成一个表格(表4.2)。

这样的表格就叫做列联表。它给出了按照两个变量交叉分类得到的每一个小类的观测个数。

为了得到这样的表格, 需要把数据输入为数据集。有时我们得到的数据是每一个观测的变量取值, 比如, 我们有每一个学生的性别(SEX)情况和来源(FROM)情况, 可以输入这些原始数据, 如:

```
data class;
  input sno sex $ from $;
  label sex='性别' from='来源';
  cards;
1  男  本地
2  女  外地
3  男  外地
···········/* 所有学生的记录 */
;
```

然后用如下的FREQ过程可以画出列联表:

```
proc freq data=class;
  tables from * sex;
run;
```

结果是下面的表格。

```
from(来源)     sex(性别)

Frequency|
Percent  |
Row Pct  |
Col Pct  |男       |女       |  Total
---------+--------+--------+
本地     |      4 |      6 |     10
         |  12.90 |  19.35 |  32.26
         |  40.00 |  60.00 |
         |  22.22 |  46.15 |
---------+--------+--------+
外地     |     14 |      7 |     21
         |  45.16 |  22.58 |  67.74
         |  66.67 |  33.33 |
         |  77.78 |  53.85 |
---------+--------+--------+
Total          18       13       31
            58.06    41.94   100.00
```

表格数据输入的另一种情况是, 我们得到的数据就已经是上面表格4.2那样的调查结果而不是具体的样本情况, 可以直接把表格输入一个数据集, 但数据集中要有一个代表观测数的变量, 例如：

```
data classt;
  input from $ sex $ numcell;
  label sex='性别' from='来源';
  cards;
本地  男  4
本地  女  6
外地  男  14
外地  女  7
;
run;
```

这样的数据要画列联表, 需要在FREQ过程中使用WEIGHT语句指定表示重复数的变量(NUMCELL):

```
proc freq data=classt;
  tables from * sex;
  weight numcell;
run;
```

结果和上例相同。

在输出结果中, 我们看到TABLES语句中的前一个变量被用来区分行, 后一个变量被用来区分列。每个格子中有四个数: Frequency(频数, 本格子的观测数), Percent(百分比), Row Pct(行百分比, 表示本类在本行中占的百分比, 比如本地男生有4个人, 本行有10个人, 占本行的40.00%), Col Pct(列百分比)。在表的右侧有行总计, 比如本地学生有10个人, 占总学生数(31人)的32.26%。在表的下侧有列总计, 比如男生有18个人, 占学生总数的58.06%。表格右下方是总数(31)和总百分比(100)。

为了作列联表, 调用FREQ过程, 使用TABLES语句指定行变量和列变量, 两者用星号分开, 如果数据本身是表格数据还需要用WEIGHT语句指定存放表格单元观测数的变量。

可以作出简化的表格, 在TABLES语句中加上NOFREQ、NOPCT、NOROW、NOCOL等选项就可以抑制相应的统计量的输出。例如, 用如下程序:

```
proc freq data=classt;
  tables from * sex / nopct norow nocol;
```

```
  weight numcell;
run;
```

就可以产生只有单元数的表格。

4.4.2 列联表独立性检验

对于数值型变量, 我们考虑其相关关系的通常的办法是计算相关系数和进行回归分析。如果我们要研究离散取值的名义变量和有序变量有无相关, 最常用的检验办法是列联表独立性检验。列联表检验的零假设是两变量X和Y相互独立, 计算一个χ^2统计量, 基于列联表中频数取值与期望取值之差, 当χ^2统计量很大时否定零假设。

例如, 为了探讨吸烟与慢性支气管炎有无关系, 调查了339人, 情况如下：

	患慢性支气管炎	未患慢性支气管炎
吸烟	43	162
不吸烟	13	121

设想有两个随机变量X, Y：X取1表示吸烟, 取2表示不吸烟, Y取1表示患慢性支气管炎, 取2表示未患。零假设为：

$$\mathrm{H}_0 : \text{X与Y相互独立}$$

要检验此零假设, 先取定检验水平0.05, 调用PROC FREQ过程, 在TABLES语句中加上CHISQ选项即可。下面的例子中还加入了EXPECTED选项以显示零假设下的期望频数值：

```
data bron;
  input smoke $ bron $ numcell;
  label smoke='吸烟'  bron='慢性支气管炎';
  cards;
吸烟  患病  43
吸烟  未患  162
不吸烟  患病  13
不吸烟  未患  121
;
proc freq data=bron;
  tables smoke*bron / nopct norow
           nocol  chisq expected;
weight numcell;
run;
```

结果如下：

```
          The FREQ Procedure

        Table of smoke by bron

  smoke(吸烟)     bron(慢性支气管炎)

  Frequency|
  Expected |患病    |未患    |  Total
  ---------+--------+--------+
  不吸烟   |     13 |    121 |    134
           | 22.136 | 111.86 |
  ---------+--------+--------+
  吸烟     |     43 |    162 |    205
           | 33.864 | 171.14 |
  ---------+--------+--------+
  Total          56      283      339
```

```
        Statistics for Table of smoke by bron

Statistic                      DF      Value      Prob
------------------------------------------------------
Chi-Square                      1     7.4688    0.0063
Likelihood Ratio Chi-Square     1     7.9250    0.0049
Continuity Adj. Chi-Square      1     6.6736    0.0098
Mantel-Haenszel Chi-Square      1     7.4467    0.0064
Phi Coefficient                      -0.1484
Contingency Coefficient               0.1468
Cramer's V                           -0.1484
```

```
       Fisher's Exact Test
----------------------------------
Cell (1,1) Frequency (F)        13
Left-sided Pr <= F          0.0041
Right-sided Pr >= F         0.9985

Table Probability (P)       0.0026
Two-sided Pr <= P           0.0069

        Sample Size = 339
```

列联表中列出了表格单元频数和在零假设下的期望频数, 可以看出, 吸烟人中患病的数目比期望数目大。检验的结果只要看后面的统计量部分的Chi-Square一行, 其值为7.4688, p值为0.0063, 所以应否定零假设, 作出结论: 吸烟与患慢性支气管炎是不独立的。

使用χ^2检验要求每个单元格期望频数不少于5。在条件不满足的时候还可以使用Fisher精确检验。对于两行两列的表格FREQ过程自动给出Fisher精确检验的结果, 其双侧检验p值为0.0069, 应拒绝零假设。

4.4.3 属性变量关联度计算

对于区间变量, 我们可以计算两两的相关系数。属性变量因为没有数值概念所以不能计算相关系数, 但对于两个有序变量我们可以计算类似于相关系数的关联性量度。其中一种关联性量度叫做Kendal Tau-b统计量, 取值在-1到1之间, 值接近于1表示正关联, 接近于-1表示负关联, 接近于0表示没有相关关系。

下面用例子说明如何在FREQ过程中计算Kendal Tau-b统计量。本例取自《SAS系统与基础统计分析》一书。假设我们要研究奶牛种群大小与其患某种细菌性疾病的关系。牛的患病程度(DISEASE)分为没有(0)、低(1)、高(2), 牛群大小(HERDSIZE)分为小(1)、中(2)、大(3)。数据如下数据步所示：

```
data cows;
  input herdsize  disease  numcell;
  label herdsize='牛群大小'
        disease='患病程度';
  cards;
1 0 9
1 1 5
1 2 9
2 0 18
2 1 4
2 2 19
3 0 11
3 1 88
3 2 136
;
run;
```

用FREQ过程在TABLES语句中加上MEASURES选项就可以计算Kendall Tau-b统计量：

```
proc freq data=cows;
  tables herdsize*disease / measures
      expected nopercent norow nocol;
  weight numcell;
  title '奶牛疾病数据分析';
run;
```

结果如下：

奶牛疾病数据分析

The FREQ Procedure

Table of herdsize by disease

herdsize(牛群大小) disease(患病程度)

Frequency| Expected	0	1	2	Total
1	9	5	9	23
	2.9231	7.4615	12.615	
2	18	4	19	41
	5.2107	13.301	22.488	
3	11	88	136	235
	29.866	76.237	128.9	
Total	38	97	164	299

```
     Statistics for Table of herdsize by disease

Statistic                                Value     ASE
-------------------------------------------------------
Gamma                                   0.4113  0.1009
Kendall's Tau-b                         0.2173  0.0606
Stuart's Tau-c                          0.1482  0.0436

Somers' D C|R                           0.2762  0.0780
Somers' D R|C                           0.1710  0.0482

Pearson Correlation                     0.2816  0.0660
Spearman Correlation                    0.2331  0.0656

Lambda Asymmetric C|R                   0.0000  0.0000
Lambda Asymmetric R|C                   0.1094  0.0794
Lambda Symmetric                        0.0352  0.0264

Uncertainty Coefficient C|R             0.0990  0.0256
Uncertainty Coefficient R|C             0.1437  0.0375
Uncertainty Coefficient Symmetric       0.1172  0.0302

                  Sample Size = 299
```

算出的Kendall Tau-b统计量值为0.2173, 渐近标准误差(ASE)为0.0606, 用统计量值加减两倍标准误差作为Kendall Tau-b的95%置信区间, 可算得(0.0961, 0.3385)在零点右边, 所以可认为奶牛患病程度与种群大小有正的关联。事实上, 我们从列联表中实际频数与期望频数的对比也可以看出, 小的种群患病比期望值轻, 大的种群患病比期望值重, 即患病程度与种群大小有正的关联。

4.4.4 用Analyst作列联表分析

Analyst的"Statistics - Table Analysis"可以进行列联表分析。比如，对吸烟危害的数据集BRON选此菜单，出现如图4.14的对话框，这里要指定行变量、列变量, 如果数据集中已经是表格则需要输入单元格

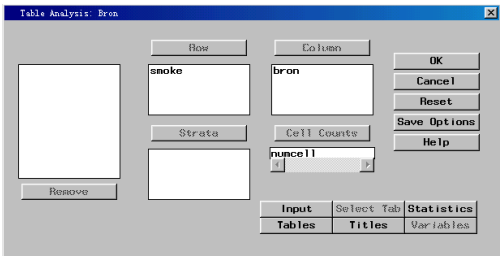


图 4.14: Analyst: Analyst: 列联表分析

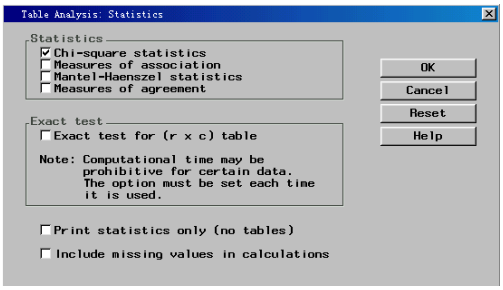


图 4.15: Analyst: Analyst: 列联表统计

计数变量(Cell Counts)。这里的Tables按钮用来指定画的表格中出现的内容，如有无各百分比、期望频数等。这里的Statistics按钮指定对列联表要作的分析，比如独立性的χ^2检验，计算关联性量度，进行精确检验，等等，见图4.15所示的对话框。如果要对上面奶牛的例子计算关联性量度就要用到这个对话框。Analyst调用PROC FREQ进行计算，所以结果与上面编程的结果相同。

练习

1. 下面是一组草原隼的鸟巢高度的数据, 试检验其分布是否正态。

   15 3.5 3.5 7 1 7 5.75 27 15 8 4.75 7.5 4.25 6.25 5.75 5 8.5 9 6.25 5.5 4 7.5 8.75 6.5 4 5.25 3 12 3.75 4.75 6.25 3.25 2.5

2. 有若干人参加了一个减肥锻炼, 在一年后测量了他们的身体脂肪含量, 结果如下(身体脂肪含量的百分数)：

男性组：13.3 19 20 8 18 22 20 31 21 12 16 12 24

女性组：22 26 16 12 21.7 23.2 21 28 30 23

比较这些人中男性和女性的身体脂肪含量有无显著差异。

3. 下表为某基础统计课程两次考试的学生成绩。两次考试考同样的知识。试比较这两次考试难易程度有无显著差异。

学号	1	2	3	4	5	6	7	8	9	10
第一次	93	88	89	88	67	89	83	94	89	55
第二次	98	74	67	92	83	90	74	97	96	81
学号	11	12	13	14	15	16	17	18	19	20
第一次	88	91	85	70	90	90	94	67	87	83
第二次	83	94	89	78	96	93	81	81	93	91

4. 为研究溶菌酶水平在患胃溃疡的病人与正常人之间有无显著差异, 测量了一组病人和一组正常人的溶菌酶水平, 结果见下表。试检验两者的溶菌酶水平有无显著差异(水平0.05)。

   胃溃疡病人组：0.2 10.4 0.3 10.9 0.4 11.3 1.1 12.4 2.0 16.2 2.1 17.6 3.3 18.9 3.8 20.7 4.5 24.0 4.8 25.4 4.9 40.0 5.0 42.2 5.3 50.0 7.5 60.0 9.8

   对照组：0.2 5.4 0.3 5.7 0.4 5.8 0.7 7.5 1.2 8.7 1.5 8.8 1.5 9.1 1.9 10.3 2.0 15.6 2.4 16.1 2.5 16.5 2.8 16.7 3.6 20.0 4.8 20.7 4.8 33.0

   提示：要考虑分布是否正态。

5. 为了考察两种测量萘含量的液体层析方法：标准方法和高压方法的测量结果有无显著差异, 取

了10份试样, 每份分为两半, 一半用标准方法测量, 一半用高压方法测量, 每个试样的两个结果如下表, 试检验这两种化验方法有无显著差异(水平0.05)：

标准	14.7	14.0	12.9	16.2	10.2	12.4	12.0	14.8	11.8	9.7
高压	12.1	10.9	13.1	14.5	9.6	11.2	9.8	13.7	12.0	9.1

6. 使用放射性金195作示踪元素注射到血液中, 下表为注射x天后血液内残留的金元素百分比y, 取了10个血样, 对数据分别拟合线性回归、负指数关系$y = Ae^{-bx}$, 并使用各种非参数曲线拟合方法拟合曲线。比较各结果。

x	1	1	2	2	2	3	5	6	6	7
y	94.5	86.4	71	80.5	81.4	67.4	49.3	46.8	42.3	36.6

7. 对数据集SASUSER.GPA中的大学学科平均成绩GPA建模, 用高中成绩HSM、HSS、HSE作为自变量。简述回归的结果。试改进模型。使用SAS/INSIGHT和REG过程两种办法。

8. 为试制某种化工产品, 在三种不同温度、四种不同压力下试验, 每一水平组合重复两次, 得到产品的收率数据如下(%)：

温度	压力			
	1	2	3	4
1	52, 57	42, 45	41, 45	48, 45
2	50, 52	47, 45	47, 48	53, 30
3	63, 58	54, 59	57, 60	58, 59

试在0.05水平下进行方差分析并简述结果。

9. 为了考察法院判决是否与被告种族有关, 调查了326位被告的判决情况：

	黑人	白人
有罪	17	19
无罪	149	141

试在0.05水平下检验判决结果与被告种族是否独立。

第五章 SAS多元统计分析

多元统计分析是统计学的重要应用工具，SAS实现了许多常用的多元统计分析方法。SAS用于多变量分析的过程有PRINCOMP(主分量分析), FACTOR(因子分析), CANCORR(典型相关分析), MDS(多维标度过程), MULTTEST(多重检验), PRINQUAL(定性数据的主分量分析), CORRESP(对应分析)，用于判别分析的过程有DISCRIM(判别分析), CANDISC(典型判别), STEPDISC(逐步判别), 用于聚类分析的过程有CLUSTER(谱系聚类), FASTCLUS(K均值快速聚类), MODECLUS(非参数聚类), VARCLUS (变量聚类), TREE (画谱系聚类的结果谱系图并给出分类结果)。我们这一章介绍最常见的多元统计方法, 更详细的资料请参考《SAS系统—SAS/STAT软件使用手册》。

5.1 多变量分析

现实中的统计对象经常用多个指标来表示, 比如人口普查, 就可以有姓名、性别、出生年月日、籍贯、婚姻状况、民族、政治面貌、地区等, 企业调查, 可以有净资产、负债、盈利、职工人数、还贷情况等

等。多个指标(变量)可以分别进行分析, 但是, 我们往往希望综合使用这些指标, 这时, 有主分量分析、因子分析等方法可以把数据的维数降低, 同时又尽量不损失数据中的信息。

5.1.1 主分量分析

一、理论介绍

主分量分析的目的是从原始的多个变量取若干线性组合, 能尽可能多地保留原始变量中的信息。从原始变量到新变量是一个正交变换(坐标变换)。设有$X = \begin{pmatrix} X_1 & \cdots & X_p \end{pmatrix}'$是一个$p$维随机变量, 有二阶矩, 记$\mu = \mathrm{E}(X)$, $\Sigma = \mathrm{Var}(X)$。考虑它的线性变换

$$\begin{cases} Y_1 = \mathbf{l}_1' X = l_{11}X_1 + \cdots + l_{p1}X_p \\ \cdots\cdots\cdots \\ Y_p = \mathbf{l}_p' X = l_{1p}X_1 + \cdots + l_{pp}X_p \end{cases}$$

易见

$$\begin{aligned} &\mathrm{Var}(Y_i) = \mathbf{l}_i'\Sigma\mathbf{l}_i \\ &\mathrm{Cov}(Y_i, Y_j) = \mathbf{l}_i'\Sigma\mathbf{l}_j \end{aligned} \qquad i, j = 1, \cdots, p$$

如果要用Y_1尽可能多地保留原始的X的信息, 经典的办法是使Y_1的方差尽可能大, 这需要对线性变换的系数$\mathbf{l}_1$加限制, 一般要求它是单位向量, 即$\mathbf{l}_1'\mathbf{l}_1 = 1$。其它的各$Y_i$也希望尽可能多地保留$X$的信息, 但前面的$Y_1, \ldots, Y_{i-1}$已保留的信息就不再保留, 即要求$\mathrm{Cov}(Y_i, Y_j) = 0$, $j = 1, \ldots, i-1$, 同时对$\mathbf{l}_i$也有$\mathbf{l}_i'\mathbf{l}_i = 1$的要求, 在这样的条件下使$\mathrm{Var}(Y_i)$最大。设协方差阵$\Sigma$的特征值为$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$, 相

应的单位特征向量分别为$a_1, a_2, \ldots, a_p$(当特征根有重根时单位特征向量不唯一)。这时X的第i个主成分为$Y_i = a_i'X$, $i = 1, \ldots, p$, 且$\mathrm{Var}(Y_i) = \lambda_i$。记$A = (\ a_1\ \cdots\ a_p\ )$, $\Lambda = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_p \end{pmatrix}$, $Y = (\ Y_1\ \cdots\ Y_p\ )$, 则$A$为正交阵, $Y = A'X$, $\mathrm{Var}(Y) = \Lambda$, 且$\Sigma_1^p \lambda_i = \Sigma_1^p \sigma_{ii}$, 其中$\sigma_{ii}$为$\Sigma$的主对角线元素。

主分量Y_k与原始变量X_i的相关系数$\rho(Y_k, X_i)$称为因子负荷量(factor loading), 可以证明$\rho(Y_k, X_i) = \sqrt{\lambda_k} a_{ik}/\sqrt{\sigma_{ii}}$, $k, i = 1, \ldots, p$, $\Sigma_{i=1}^p \sigma_{ii}\rho^2(Y_k, X_i) = \lambda_k$, $\Sigma_{k=1}^p \rho^2(Y_k, X_i) = \Sigma_{k=1}^p \lambda_k a_{ik}^2/\sigma_{ii} = 1$。

为了减少变量的个数, 希望前几个Y_i就可以代表X的大部分信息。定义$\lambda_k/\Sigma_1^p \lambda_i$为主分量$Y_k$的贡献率, 称$\Sigma_{i=1}^m \lambda_i/\Sigma_{i=1}^p \lambda_i$为主分量$Y_1, \ldots, Y_m$的累计贡献率。一般取$m$使得累计贡献率达到70%—80%以上。累计贡献率表示m个主分量从$X_1, \ldots, X_p$中提取了多少信息, 但没有表达用它来恢复每一个X_i能恢复多少, 为此定义m个主分量$Y_1, \ldots, Y_m$对原始变量X_i的贡献率ν_i, ν_i为X_i对$Y_1, \ldots, Y_m$的复相关系数平方, 可以用公式$\nu_i = \Sigma\, \lambda_k a_{ik}^2/\sigma_{ii}$计算(注意$m = p$时$\nu_i \equiv 1$)。前$m$个主分量$Y_{(m)} = (\ Y_1\ \cdots\ Y_m\ )'$在$X$的$m$个线性组合中能对$X$最好地线性逼近。

在上面的主分量计算方法中, 方差越大的变量越被优先保留信息, 实际中为了消除这种影响经常把变量标准化, 即令

$$X_i^* = \frac{X_i - \mathrm{E}X_i}{\sqrt{\mathrm{Var}(X_i)}}, i = 1, \ldots, p$$

这时$X^* = (X_1^* \cdots X_p^*)$的协方差阵就是$X$的相关阵$R$。这时, 主分量的协方差阵是$\Lambda^* = \mathrm{diag}(\lambda_1^*, \ldots, \lambda_p^*)$, 其中$\lambda_1^* \geq \cdots \geq \lambda_p^*$为$R$的特征根；$\Sigma_{i=1}^p \lambda_i^* = p$；$X_i^*$与主分量$Y_k^*$的相关系数(因子负荷量)为$\rho(Y_k^*, X_i^*) = \sqrt{\lambda_k^*} a_{ik}^*$, 其中$a_{ik}^* = (a_{1k}^*, \ldots, a_{pk}^*)$为$R$的对应$\lambda_k^*$的单位特征向量；$\Sigma_{i=1}^p \rho^2(Y_k^*, X_i^*) = \lambda_k^*$；$\Sigma_{k=1}^p \rho^2(Y_k^*, X_i^*) = 1$。

对于X的观测样本, 设第t次观测为$x_{(t)} = (x_{t1}, \ldots, x_{tp}), t = 1, \ldots, n,$ 把数据写成矩阵形式为

$$\tilde{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}$$

由$\tilde{X}$得协方差阵Σ的估计$\hat{\Sigma}$和相关阵R的估计$\hat{R}$, 从$\hat{\Sigma}$或$\hat{R}$可以得到主分量分解。计算特征值和单位特征向量, 仍记为$\lambda_1 \geq \lambda_2 \geq \cdots \geq = \lambda_p$和$a_1, a_2, \ldots, a_p$, 用$Y_i = a_i' X$作为$X$的第$i$主分量, 而$Y_{(t)} = x_{(t)} A$称为第t个观测的主分量得分。结果得到的主分量得分矩阵为$\tilde{Y} = XA$。可以把$\tilde{Y}$的前几列作为维数压缩后的数据。在SAS的PRINCOMP中计算主分量得分时如果主分量分解是用相关阵得到的则原始自变量要先标准化(减去均值并除以标准差), 如果分解用的是协方差阵则计算主分量得分时只对原始变量中心化(减去均值)。

二、用PRINCOMP过程计算主分量分析

SAS的PRINCOMP过程有如下功能：

- 完成主分量分析。
- 主分量的个数可以由用户自己确定, 主分量的名字可以用户自己规定, 主分量得分是否标准化可自己规定。
- 输入数据集可以是原始数据集、相关阵、协方差阵或叉积阵。输入为原始数据时, 用户还可以规定从协方差阵出发还是从相关阵出发进行分析。由协方差阵出发时方差大的变量在分析中起到更大的作用。
- 计算结果有：简单统计量, 相关阵或协方差阵, 从大到小排序的特征值和相应特征向量, 每个主分量解释的方差比例, 累计比例等。可生成两个输出数据集：一个包含原始数据及主分量得分, 另一个包含有关统计量, 类型为TYPE=CORR或COV。
- 可揭示变量间的共线关系。若某特征值特别接近0说明变量线性相关, 这时用这些变量作回归自变量可能得到错误的结果。

PRINCOMP过程主要使用PROC PRINCOMP语句与VAR语句。PROC PRINCOMP语句用来规定输入输出和一些运行选项, 包括：

- DATA=输入数据集, 可以是原始数据集, 也可以是TYPE=CORR,COV的数据集
- OUT=输出包含原始数据和主分量得分的数据集
- OUTSTAT=统计量输出数据集

- COV 要求从协方差阵出发计算主分量。缺省为从相关阵出发计算。
- N=要计算的主分量个数。缺省时全算。
- NOINT 要求在模型中不使用截距项。这时统计量输出数据集类型为TYPE=UCORR或UCOV。
- STD 要求在OUT=的数据集中把主分量得分标准化为单位方差。不规定时方差为相应特征值。

用VAR语句指定原始变量, 必须为数值型(区间变量)。

三、应用举例

例1. 一月和七月平均气温的主分量分析

在数据集TEMPERAT中存放有美国一些城市一月和七月的平均气温。我们希望对这两个气温进行主成分分析, 希望用一个统一的温度来作为总的可比的温度, 所以进行主分量分析。程序如下：

```
DATA TEMPERAT;
   INPUT CITY $1-15 JANUARY JULY;
   CARDS;
MOBILE           10.7  27.6
PHOENIX          10.7  32.9
LITTLE ROCK       4.2  27.4
SACRAMENTO        7.3  24.0
DENVER           -1.2  22.8
HARTFORD         -4.0  22.6
WILMINGTON        0.0  24.3
WASHINGTON DC     2.0  25.9
JACKSONVILLE     12.6  27.2
```

MIAMI	19.6	27.9
ATLANTA	5.8	25.6
BOISE	-1.7	23.6
CHICAGO	-5.1	22.2
PEORIA	-4.6	23.9
INDIANAPOLIS	-2.3	23.9
DES MOINES	-7.0	23.9
WICHITA	-0.4	27.1
LOUISVILLE	0.7	24.9
NEW ORLEANS	11.6	27.7
PORTLAND, MAINE	-5.8	20.0
BALTIMORE	0.8	24.8
BOSTON	-1.6	22.9
DETROIT	-3.6	22.9
SAULT STE MARIE	-9.9	17.7
DULUTH	-13.1	18.7
MINNEAPOLIS	-11.0	22.2
JACKSON	8.4	27.6
KANSAS CITY	-2.3	26.0
ST LOUIS	-0.4	25.9
GREAT FALLS	-6.4	20.7
OMAHA	-5.2	25.1
RENO	-0.1	20.7
CONCORD	-6.3	20.9
ATLANTIC CITY	0.4	23.9
ALBUQUERQUE	1.8	25.9
ALBANY	-5.8	22.2
BUFFALO	-4.6	21.2
NEW YORK	0.1	24.8
CHARLOTTE	5.6	25.8
RALEIGH	4.7	25.3
BISMARCK	-13.2	21.6
CINCINNATI	-0.5	24.2
CLEVELAND	-2.8	21.9
COLUMBUS	-2.0	23.1
OKLAHOMA CITY	2.7	27.5
PORTLAND, OREG	3.4	19.5
PHILADELPHIA	0.2	24.9
PITTSBURGH	-2.2	22.2
PROVIDENCE	-2.0	22.3
COLUMBIA	7.4	27.3

```
SIOUX FALLS      -9.9  22.9
MEMPHIS           4.7  26.4
NASHVILLE         3.5  26.4
DALLAS            7.1  29.3
EL PASO           6.4  27.9
HOUSTON          11.2  28.5
SALT LAKE CITY   -2.2  24.8
BURLINGTON       -8.4  21.0
NORFOLK           4.7  25.7
RICHMOND          3.1  25.5
SPOKANE          -3.7  20.9
CHARLESTON, WV    1.4  23.9
MILWAUKEE        -7.0  21.1
CHEYENNE         -3.0  20.6
;
PROC PRINCOMP COV OUT=PRIN;
  VAR JULY JANUARY;
RUN;
```

主成分得分输出到了数据集PRIN中，显示的输出结果如下：

```
           The PRINCOMP Procedure
          Observations          64
          Variables              2

             Simple Statistics
                    JULY           JANUARY
  Mean       24.22656250       0.052951389
  StD         2.84867728       6.506907270

             Covariance Matrix
                     JULY           JANUARY
JULY           8.11496225       14.45317629
JANUARY       14.45317629       42.33984222

       Total Variance    50.454804466

    Eigenvalues of the Covariance Matrix
```

```
  Eigenvalue  Difference  Proportion  Cumulative
1 47.6267305  44.7986565      0.9439      0.9439
2  2.8280740                  0.0561      1.0000

                Eigenvectors
                     Prin1         Prin2
      JULY         0.343532      0.939141
      JANUARY      0.939141      -.343532
```

结果包括观测数、变量数，分析变量的简单统计量、协方差阵，总的方差，协方差阵的特征值(包括特征值、特征值之间的差、比例、累计比例)，特征向量。程序中的COV选项是要求用协方差阵而不是相关阵的特征值，结果中给了特征向量，我们就可以知道主分量得分的计算公式为：

```
PRIN1 = 0.343532 * (JULY - 24.23) + 0.939141 * (JANUARY - 0.05)
PRIN2 = 0.939141 * (JULY - 24.23) - 0.343532 * (JANUARY - 0.05)
```

如果没有用COV选项, 原始变量中心化后还需要除以标准差。由系数可见, 第一主分量是两个月份的加权平均, 代表了一个地方的气温水平, 第二主分量系数一正一负, 反应了冬季和夏季的气温差别。

在INSIGHT中打开WORK.PRIN, 分别绘制JULY对JANUARY、PRIN2 对PRIN1 的散点图(图5.1)。从图可以看出主分量为原始变量的一个正交旋转。

例2．美国各种类型犯罪的主分量分析

在数据集CRIME中有美国各个州的各种类型犯罪的犯罪率数据。希望对这些犯罪率数据进行主分量分析以概括犯罪情况。程序如下：

```
DATA CRIME;
   TITLE '各州每十万人的犯罪率';
```

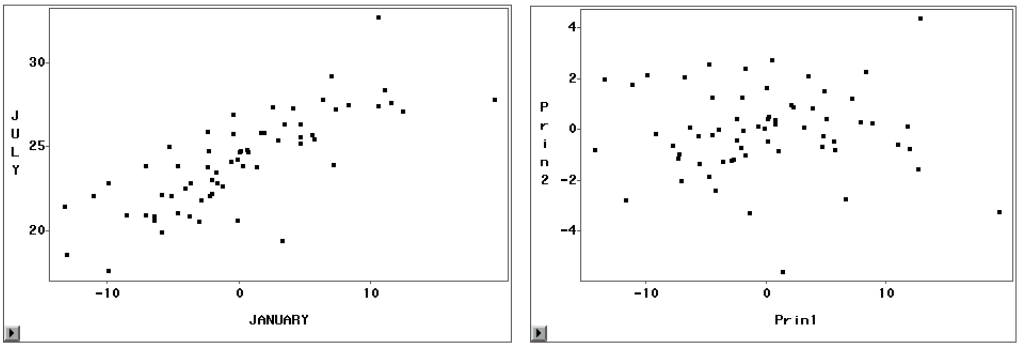


图 5.1: 一月、七月气温的散点图和主分量的散点图

```
   INPUT STATE $1-15 MURDER RAPE ROBBERY ASSAULT
         BURGLARY LARCENY AUTO;
   CARDS;
ALABAMA        14.2 25.2  96.8 278.3 1135.5 1881.9 280.7
ALASKA         10.8 51.6  96.8 284.0 1331.7 3369.8 753.3
ARIZONA         9.5 34.2 138.2 312.3 2346.1 4467.4 439.5
ARKANSAS        8.8 27.6  83.2 203.4  972.6 1862.1 183.4
CALIFORNIA     11.5 49.4 287.0 358.0 2139.4 3499.8 663.5
COLORADO        6.3 42.0 170.7 292.9 1935.2 3903.2 477.1
CONNECTICUT     4.2 16.8 129.5 131.8 1346.0 2620.7 593.2
DELAWARE        6.0 24.9 157.0 194.2 1682.6 3678.4 467.0
FLORIDA        10.2 39.6 187.9 449.1 1859.9 3840.5 351.4
GEORGIA        11.7 31.1 140.5 256.5 1351.1 2170.2 297.9
HAWAII          7.2 25.5 128.0  64.1 1911.5 3920.4 489.4
IDAHO           5.5 19.4  39.6 172.5 1050.8 2599.6 237.6
ILLINOIS        9.9 21.8 211.3 209.0 1085.0 2828.5 528.6
INDIANA         7.4 26.5 123.2 153.5 1086.2 2498.7 377.4
IOWA            2.3 10.6  41.2  89.8  812.5 2685.1 219.9
KANSAS          6.6 22.0 100.7 180.5 1270.4 2739.3 244.3
KENTUCKY       10.1 19.1  81.1 123.3  872.2 1662.1 245.4
LOUISIANA      15.5 30.9 142.9 335.5 1165.5 2469.9 337.7
MAINE           2.4 13.5  38.7 170.0 1253.1 2350.7 246.9
MARYLAND        8.0 34.8 292.1 358.9 1400.0 3177.7 428.5
MASSACHUSETTS   3.1 20.8 169.1 231.6 1532.2 2311.3 1140.1
MICHIGAN        9.3 38.9 261.9 274.6 1522.7 3159.0 545.5
MINNESOTA       2.7 19.5  85.9  85.8 1134.7 2559.3 343.1
MISSISSIPPI    14.3 19.6  65.7 189.1  915.6 1239.9 144.4
MISSOURI        9.6 28.3 189.0 233.5 1318.3 2424.2 378.4
MONTANA         5.4 16.7  39.2 156.8  804.9 2773.2 309.2
NEBRASKA        3.9 18.1  64.7 112.7  760.0 2316.1 249.1
NEVADA         15.8 49.1 323.1 355.0 2453.1 4212.6 559.2
NEW HAMPSHIRE   3.2 10.7  23.2  76.0 1041.7 2343.9 293.4
NEW JERSEY      5.6 21.0 180.4 185.1 1435.8 2774.5 511.5
```

```
NEW MEXICO      8.8 39.1 109.6 343.4 1418.7 3008.6 259.5
NEW YORK       10.7 29.4 472.6 319.1 1728.0 2782.0 745.8
NORTH CAROLINA 10.6 17.0  61.3 318.3 1154.1 2037.8 192.1
NORTH DAKOTA    0.9  9.0  13.3  43.8  446.1 1843.0 144.7
OHIO            7.8 27.3 190.5 181.1 1216.0 2696.8 400.4
OKLAHOMA        8.6 29.2  73.8 205.0 1288.2 2228.1 326.8
OREGON          4.9 39.9 124.1 286.9 1636.4 3506.1 388.9
PENNSYLVANIA    5.6 19.0 130.3 128.0  877.5 1624.1 333.2
RHODE ISLAND    3.6 10.5  86.5 201.0 1489.5 2844.1 791.4
SOUTH CAROLINA 11.9 33.0 105.9 485.3 1613.6 2342.4 245.1
SOUTH DAKOTA    2.0 13.5  17.9 155.7  570.5 1704.4 147.5
TENNESSEE      10.1 29.7 145.8 203.9 1259.7 1776.5 314.0
TEXAS          13.3 33.8 152.4 208.2 1603.1 2988.7 397.6
UTAH            3.5 20.3  68.8 147.3 1171.6 3004.6 334.5
VERMONT         1.4 15.9  30.8 101.2 1348.2 2201.0 265.2
VIRGINIA        9.0 23.3  92.1 165.7  986.2 2521.2 226.7
WASHINGTON      4.3 39.6 106.2 224.8 1605.6 3386.9 360.3
WEST VIRGINIA   6.0 13.2  42.2  90.9  597.4 1341.7 163.3
WISCONSIN       2.8 12.9  52.2  63.7  846.9 2614.2 220.7
WYOMING         5.4 21.9  39.7 173.9  811.6 2772.2 282.0
;
PROC PRINCOMP OUT=CRIMCOMP;
RUN;

PROC SORT;
   BY PRIN1;
PROC PRINT;
   ID STATE;
   VAR PRIN1 PRIN2 MURDER RAPE ROBBERY ASSAULT
       BURGLARY LARCENY AUTO;
   TITLE2 '各州按第一主分量作为总犯罪率排列';
PROC SORT;
   BY PRIN2;
PROC PRINT;
   ID STATE;
   VAR PRIN1 PRIN2 MURDER RAPE ROBBERY ASSAULT
       BURGLARY LARCENY AUTO;
   TITLE2 '各州按第二主分量作为金钱犯罪与暴力犯罪对比的排列';
GOPTIONS FTEXT='宋体';
PROC GPLOT;
   PLOT PRIN2*PRIN1=STATE;
   TITLE2 '前两个主分量的散点图';
PROC GPLOT;
   PLOT PRIN3*PRIN1=STATE;
   TITLE2 '第一、三主分量的散点图';
RUN;
```

输入数据后, 用PROC PRINCOMP对数据进行主分量分析, 结果先给出了各变量的简单统计量, 变量的相关阵, 其特征值和特征向量结果如下：

Eigenvalues of the Correlation Matrix

	Eigenvalue	Difference	Proportion	Cumulative
1	4.11495951	2.87623768	0.5879	0.5879
2	1.23872183	0.51290521	0.1770	0.7648
3	0.72581663	0.40938458	0.1037	0.8685
4	0.31643205	0.05845759	0.0452	0.9137
5	0.25797446	0.03593499	0.0369	0.9506
6	0.22203947	0.09798342	0.0317	0.9823
7	0.12405606		0.0177	1.0000

Eigenvectors

	Prin1	Prin2	Prin3	Prin4
MURDER	0.300279	-.629174	0.178245	-.232114
RAPE	0.431759	-.169435	-.244198	0.062216
ROBBERY	0.396875	0.042247	0.495861	-.557989
ASSAULT	0.396652	-.343528	-.069510	0.629804
BURGLARY	0.440157	0.203341	-.209895	-.057555
LARCENY	0.357360	0.402319	-.539231	-.234890
AUTO	0.295177	0.502421	0.568384	0.419238

Eigenvectors

	Prin5	Prin6	Prin7
MURDER	0.538123	0.259117	0.267593
RAPE	0.188471	-.773271	-.296485
ROBBERY	-.519977	-.114385	-.003903
ASSAULT	-.506651	0.172363	0.191745
BURGLARY	0.101033	0.535987	-.648117
LARCENY	0.030099	0.039406	0.601690
AUTO	0.369753	-.057298	0.147046

第一主分量贡献率只有59%, 前两个主分量累计贡献率达到76%, 可以用前两个主分量。前三个主分量累计贡献率已达到87%, 所以前三个主分量可以表现犯罪率的大部分信息。第一主分量的计算系数都是正数, 所以它是一个州的犯罪率的一个加权平均, 代表这个州的总的犯罪情况。第二主分量在入室盗窃(BURGLARY)、盗窃罪(LARCENY)、汽车犯罪(AUTO)上有较大的正系数, 在谋杀(MURDER)、强奸(RAPE)、斗殴(ASSAULT)上有较大的负系数, 所以代表了暴力犯罪与其它犯罪的一种对比。第三主分量为抢劫、汽车犯罪等与盗窃罪、入室盗窃、强奸的对比, 其意义不易解释。

为了看出各州按第一主分量和第二主分量由低到高排列的情况, 先用SORT过程排了序, 然后用PRINT过程打印了结果(结果略)。在按第一主分量排序中, North Dakota、South Dakota、West Virginia排列在前, 说明其犯罪率最低, Nevada、California排列在后, 说明其犯罪率最高。在按第二主分量排列的结果中, Mississippi排在最前, 说明其暴力犯罪最高, Massachusetts最后, 说明其暴力犯罪最低。后面用PLOT过程画了主成分的散点图。

四、用SAS / INSIGHT 和Analyst 进行主分量分析

在SAS/INSIGHT中可进行主分量分析。例如, 对于上面的WORK.CRIME数据集, 在INSIGHT中打开它后, 选“Analyze - Multivariate (Y's)”, 弹出选择变量的对话框, 把各犯罪率变量都选为Y变量, 然后

按Output按钮, 选中主分量分析(Principal Component Analysis)复选框, OK后就得到了多变量分析结果(包括原始变量的简单统计量、相关阵)和主分量分析的结果(特征值、累计贡献率、特征向量)。对话框的Method按钮可以选是按协方差阵计算还是按相关阵计算。另外还画了前两个主分量的散点图。

在Analyst 中也可以进行主分量分析。选"Statistics - Multivariate - Principal Components" 可以打开主分量分析的对话框，可以选要分析的变量, 其Statistics按钮可以选按协方差阵分析还是按相关阵分析。此界面调用PROC PRINCOMP, 所以得到的结果与前面的例子是一样的。

5.1.2 因子分析

一、理论简介

主分量分析作p个原始变量的m个线性组合, 这些线性组合在原始变量的所有m个线性组合中可以最好地预报原始变量。因子分析对主分量分析进行了推广, 它用潜在的m个"因子"来概括原始变量的信息, 这些因子不一定是原始变量的线性组合。

设$\mathbf{x}$为$p \times 1$随机向量, 其均值为μ, 协方差阵为$\Sigma = (\sigma_{ij})$, 我们称$\mathbf{x}$满足有k个因子的模型, 若$\mathbf{x}$能表为

$$\mathbf{x} = \mu + \Lambda\mathbf{f} + \mathbf{u}$$

其中$\Lambda : p \times k$是未知常数阵, $\mathbf{f} : k \times p$和$\mathbf{u} : p \times 1$为随机向量。$\mathbf{f}$称为公共因子, $\mathbf{u}$叫做特殊因子, Λ叫因子载荷矩阵。这个模型象是回归分析模型, 但是这

里$\mathbf{x}$是多元随机变量而不是一个随机变量的样本, $\mathbf{f}$也是随机变量而不是一般的回归系数。求因子分解要用到原始变量协方差阵Σ。Σ与Λ和特殊因子的协方差阵Ψ的如下关系式：

$$\Sigma = \Lambda\Lambda' + \Psi$$

公因子模型分解是不唯一的, 因为如果Γ是一个正交阵, 则有

$$\mathbf{x} = \mu + (\Lambda\Gamma)(\Gamma'\mathbf{f}) + \mathbf{u}$$

这时$\Gamma'\mathbf{f}$是新的公共因子, $\Lambda\Gamma$是新的因子载荷阵。我们可以利用这一特点对得到的因子模型进行旋转以产生容易解释的因子。旋转时一般试图使因子载荷系数靠近正负1和0, 这样容易解释因子的组成。

二、FACTOR过程使用

SAS/STAT的FACTOR过程可以进行因子分析、分量分析和因子旋转。对因子模型可以使用正交旋转和斜交旋转, 可以用回归法计算得分系数, 同时把因子得分的估计存贮在输出数据集中；用FACTOR过程计算的所有主要统计量也能存贮在输出数据集中。

FACTOR过程用法很简单, 主要使用如下语句：

```
PROC FACTOR DATA= 数据集  选项;
      VAR 原始变量;
RUN;
```

输出结果包括特征值情况、因子载荷、公因子解释比例, 等等。为了计算因子得分, 一般在PROC FACTOR语句中加一个SCORE 选项和“OUTSTAT=输出数据集”选项, 然后用如下的得分过程计算公因子得分：

```
PROC SCORE DATA=原始数据集
           SCORE=FACTOR过程的输出数据集
           OUT=得分输出数据集;
     VAR 用来计算得分的原始变量集合;
RUN;
```

三、例子

数据集SOCECON为洛杉基12个地区统计的五个社会经济指标：人口总数(POP), 教育程度(SCHOOL), 就业数(EMPLOY), 服务业人数(SERVICES), 中等的房价(HOUSE)。用FACTOR过程可以进行主分量分析。下例中的SIMPLE选项要求计算变量的简单统计量, CORR要求输出相关阵。

```
DATA SOCECON;
   TITLE '五个经济指标的分析';
   INPUT POP SCHOOL EMPLOY SERVICES HOUSE;
   CARDS;
5700     12.8    2500    270     25000
1000     10.9    600     10      10000
3400     8.8     1000    10      9000
3800     13.6    1700    140     25000
4000     12.8    1600    140     25000
8200     8.3     2600    60      12000
1200     11.4    400     10      16000
9100     11.5    3300    60      14000
```

```
9900     12.5    3400    180     18000
9600     13.7    3600    390     25000
9600     9.6     3300    80      12000
9400     11.4    4000    100     13000
;
PROC FACTOR DATA=SOCECON SIMPLE CORR;
   TITLE2 '主分量分析';
RUN;
```

结果给出了五个变量的简单统计量, 相关阵, 和相关阵的特征值、累计贡献：

```
Eigenvalues of the Correlation Matrix: Total = 5  Average = 1

      Eigenvalue  Difference  Proportion  Cumulative

   1  2.87331359  1.07665350      0.5747      0.5747
   2  1.79666009  1.58182321      0.3593      0.9340
   3  0.21483689  0.11490283      0.0430      0.9770
   4  0.09993405  0.08467868      0.0200      0.9969
   5  0.01525537                 0.0031      1.0000

   2 factors will be retained by the MINEIGEN criterion.
```

前两个主分量解释了93.4%的方差, 按照缺省的选择因子个数的准则MINEIGEN, 取大于1的特征值, 所以取两个因子。因子模式阵(factor pattern, 或称因子载荷阵)为最重要的结果之一：

```
            Factor Pattern

                Factor1      Factor2

    POP         0.58096      0.80642
    SCHOOL      0.76704     -0.54476
    EMPLOY      0.67243      0.72605
    SERVICES    0.93239     -0.10431
    HOUSE       0.79116     -0.55818
```

它们是用公因子预报原始变量的回归系数。第一主分量(因子)在所有五个变量上都有正的载荷, 可见这个因子反应了城市规模的影响。第二主分量在人口、就业上有大的正载荷, 在教育程度和住房价格上有大的负载荷, 则第二个因子较大的城市人口多但是教育程度和住房价格低。结果还给出了公因子解释能力的估计：

```
      Final Communality Estimates: Total = 4.669974

       POP       SCHOOL      EMPLOY     SERVICES       HOUSE

0.98782629  0.88510555  0.97930583  0.88023562  0.93750041
```

这里给出了公因子对每一个原始变量的解释能力的量度, 这是用原始变量对公因子的复相关系数平方(取0到1间值)来计算的。Communality Estimates: Total是这些复相关系数平方的总和。因为每一个复相关系数平方都比较大, 所以我们可以认为两个公因子可以很好地解释原始变量中的信息。但是我们得到的因子解释不够清楚, 于是考虑用其它的因子分析方法。

我们来进行主因子分析。用FACTOR过程作主因子分析与作主分量分析的不同只是增加一个PRIORS=选项, 可以用PRIORS＝SMC或者MAX、ONE等。例如：

```
PROC FACTOR DATA=SOCECON priors=smc;
  TITLE2 '主因子分析';
RUN;
```

主因子法计算简约了的相关阵(相当于$\Sigma - \Psi$的估计)的特征值, 所以其特征值可能为负值。选取

因子个数的缺省准则是PROPORTION=1, 即累计特征值达到特征值总和的100%。这样取了两个因子。结果与主分量分析相似。为了得到好的因子解释, 我们在上面的PROC FACTOR语句中再加上一个ROTATE=PROMAX旋转选项, 这样将在得到主因子分析后先产生方差最大正交预旋转(VARIMAX)然后进行斜交旋转, 并加了一个REORDER选项使输出时把原始变量受相同因子影响的放在一起：

```
PROC FACTOR DATA=SOCECON PRIORS=SMC
            ROTATE=PROMAX REORDER;
  TITLE2 '主因子分析及PROMAX斜交旋转';
RUN;
```

在初始的主因子结果之后是方差最大预旋转的结果(只显示了旋转阵和旋转后的因子载荷)：

```
     Orthogonal Transformation Matrix
                      1             2
         1       0.78895       0.61446
         2      -0.61446       0.78895

           Rotated Factor Pattern
                   Factor1       Factor2
    HOUSE          0.94072      -0.00004
    SCHOOL         0.90419       0.00055
    SERVICES       0.79085       0.41509
    POP            0.02255       0.98874
    EMPLOY         0.14625       0.97499
```

可见第一因子反映了房价、教育水平、服务业人数, 这些应该与发达程度有关。第二因子反映了人口和就业情况, 与城市规模有关。这样得到的因子已经比较好用。

我们再看斜交旋转的结果, 这里只给出了旋转后的因子载荷阵：

```
          Rotated Factor Pattern
  (Standardized Regression Coefficients)
                   Factor1       Factor2
    HOUSE        0.95558485    -0.0979201
    SCHOOL       0.91842142    -0.0935214
    SERVICES     0.76053238    0.33931804
    POP          -0.0790832    1.00192402
    EMPLOY          0.04799    0.97509085
```

从结果看得到的因子比正交旋转没有改进。因为斜交旋转后的公因子是相关的, 所以结果中还给出了公因子的相关阵, 参考结构(Reference Structure, 为每个原始变量与公因子扣除其它公因子影响的偏相关), 因子结构(Factor Structure, 为原始变量与公因子间的相关系数)。

为了产生因子得分, 需要在FACTOR过程中使用SCORE 选项和OUTSTAT= 选项输出得分系数数据集并调用SCORE 过程。比如, 为了计算方差最大正交旋转的主因子得分, 可以用如下程序：

```
PROC FACTOR DATA=SOCECON PRIORS=SMC
     ROTATE=VARIMAX REORDER SCORE OUTSTAT=OUTF;
  TITLE2 '主因子分析及VARIMAX正交旋转';
RUN;

PROC SCORE DATA=SOCECON SCORE=OUTF OUT=OUTS;
  TITLE2 ' VARIMAX正交旋转后的主因子得分';
RUN;
```

5.2 判别分析

判别分析的目的是对已知分类的数据建立由数值指标构成的分类规则, 然后把这样的规则应用到未知分类的样本去分类。例如, 我们有了患胃炎的病人和健康人的一些化验指标, 就可以从这些化验指标发现两类人的区别, 把这种区别表示为一个判别公式, 然后对怀疑患胃炎的人就可以根据其化验指标用判别公式诊断。

5.2.1 统计背景

判别分析的方法有参数方法和非参数方法。参数方法假定每个类的观测来自(多元)正态分布总体, 各类的分布的均值(中心)可以不同。非参数方法不要求知道各类所来自总体的分布, 它对每一类使用非参数方法估计该类的分布密度, 然后据此建立判别规则。

记X为用来建立判别规则的p维随机变量, S为合并协方差阵估计, $t_1, \ldots, G$为组的下标, 共有G个组。记n_t为第t组中训练样本的个数, m_t为第t组的自变量均值向量, S_t为第t组的协方差阵, $|S_t|$为S_t的行列式, q_t为第t组出现的先验概率, $p(t|x)$为自变量等于x的观测属于第t组的后验概率, $f_t(x)$为第t组的分布密度在$X = x$处的值, $f(x)$为非条件密度$\Sigma_{i=1}^{G} q_t f_t(x)$。

按照Bayes理论, 自变量为x的观测属于第t组的后验概率$p(t|x) = q_t f_t(x)/f(x)$。于是, 可以把自变量X的取值空间R^p划分为G个区域$R_t, t = 1, \ldots, G$, 使得当X的取值x属于R_t时后验概率在第t组最大, 即

$$p(t|x) = \max_{s=1,\ldots,G} p(s|x), \forall x \in R_t$$

建立的判别规则为：计算自变量x到每一个组中心的广义平方距离, 并把x判入最近的类。广义平方距离的计算可能使用合并的协方差阵估计或者单独的协方差阵估计, 并与先验概率有关, 定义为

$$D_t^2(x) = d_t^2(x) + g_1(t) + g_2(t)$$

其中

$$\begin{aligned}
d_t^2(x) &= (x - m_t)' V_t^{-1} (x - m_t) \\
g_1(t) &= \begin{cases} \ln |S_t| & \text{采用单类的协方差阵估计} \\ 0 & \text{采用合并协方差阵估计} \end{cases} \\
g_2(t) &= \begin{cases} \ln q_t & \text{各类先验概率不等} \\ 0 & \text{各类先验概率相等} \end{cases}
\end{aligned}$$

$V_t = S_t$(使用单个类的协方差阵估计)或$V_t = S$(使用合并的协方差阵估计)。m_t可以用第t组的均值$\bar{X}_t$代替。在使用合并协方差阵时,

$$\begin{aligned}
D_t^2(x) &= (x - \bar{X}_t)' S^{-1} (x - \bar{X}_t) - 2 \ln q_t \\
&= x' S^{-1} x + (\bar{X}' S^{-1} \bar{X} - 2 \ln q_t) - 2 x' S^{-1} \bar{X}_t
\end{aligned}$$

其中$x'S^{-1}x$是共同的可以不考虑, 于是在比较x到各组中心的广义平方距离时, 只要计算线性判别函数$\tilde{D}_t^2(x) = \left(-\frac{1}{2}\bar{X}_t' S^{-1} \bar{X}_t + \ln q_t\right) + x' S^{-1} \bar{X}_t$, 当$x$到

第t组的线性判别函数最大时把x对应观测判入第t组。如果使用单个类的协方差阵估计$V_t = S_t$则距离函数是x的二次函数, 称为二次判别函数。

后验概率可以用广义距离表示为

$$p(t|x) = \frac{e^{-\frac{1}{2}D_t^2(x)}}{\sum_{u=1}^{G} e^{-\frac{1}{2}D_u^2(x)}}$$

因此, 参数方法的判别规则为：先决定是使用合并协方差阵还是单个类的协方差阵, 计算x到各组的广义距离, 把x判入最近的组；或者计算x属于各组的后验概率, 把x判入后验概率最大的组。如果x的最大的后验概率都很小(小于一个给定的界限), 则把它判入其它组。

非参数判别方法仍使用Bayes后验概率密度的大小来进行判别, 但这时第t组在x处的密度值$f_t(x)$不再具有参数形式, 不象参数方法那样可以用m_t和S_t(或S)表示出来。非参数方法用核方法或最近邻方法来估计概率密度$f_t(x)$。

最近邻估计和核估计也都需要定义空间中的距离。除了可以用欧氏距离外, 还可以用马氏(Mahalanobis)距离, 定义为：

$$d_t^2(x, y) = (x - y)' V_t^{-1} (x - y)$$

其中V_t为以下形式之一：

$V_t = S$		合并协方差阵
$V_t = \mathrm{diag}(S)$		合并协方差阵的对角阵
$V_t = S_t$		第t组内的协方差阵
$V_t = \mathrm{diag}S_t$		第t组的协方差阵的对角阵
$V_t = I$		单位阵，这时距离即普通欧式距离

5.2.2 DISCRIM过程的语句说明

SAS/STAT的DISCRIM过程可以进行参数判别分析和非参数判别分析, 其一般格式如下：

```
PROC DISCRIM DATA=输入数据集;
    CLASS 分类变量;
    VAR 判别用自变量集合;
RUN;
```

其中, PROC DISCRIM语句的选项中“输入数据集”为训练数据的数据集, 包括一个分类变量(在CLASS语句中说明)和用来建立判别公式的自变量集合(在VAR语句中说明)。可以用“TESTDATA=数据集”选项指定一个检验数据集, 检验数据集必须包含与训练数据集相同的自变量集合, 用训练数据集产生判别规则后将对检验数据集中的每一个观测给出分类值, 如果这个检验数据集中有表示真实分类的变量可以在过程中用“TESTCLASS 分类变量”语句指定, 这样可以检验判别的效果如何。用“OUTSTAT=数据集”指定输出判别函数的数据集, 后面可以再次用DISCRIM过程把这样输出的判别函数作为输入数据集(DATA=)读入并用它来判别检验数据(TESTDATA=)。用“OUT=数据集”指定存放训练样本及后验概率、交叉确认分类的数据集。用“OUTD=数据集”指定存放训练样本及分组的密度估计的数据集。用“TESTOUT=数据集”指定存放检验数据的后验概率及分类结果的数据集。用“TESTOUTD=数据集”输出检验数据及分组密度估计。

PROC DISCRIM语句还有一些指定判别分析方法的选项。用METHOD=NORMAL或NPAR选择参数方法或非参数方法。用POOL=NO或TEST或YES表示不用合并协方差阵、通过检验决定是否使用合并协方差阵、用合并协方差阵。如果使用非参数方法, 需要指定“R=核估计半径”选项来规定核估计方法或者指定“K=最近邻个数”来规定最近邻估计方法。

PROC DISCRIM语句有一些规定显示结果的选项。用LISTERR显示训练样本错判的观测。用CROSLISTERR显示用交叉核实方法对训练样本判别错判的观测。用LIST对每一观测显示结果。用NOCLASSIFY取消对训练样本的分类检验。用CROSSLIST显示对训练样本的交叉核实的判别结果。用CROSSVALIDATE要求进行交叉核实。当有用“TESTDATA=”指定的检验数据集时用TESTLIST选项显示检验数据集的检验结果, 当有TESTCLASS语句时用TESTLISTERR可以列出检验样本判错的观测, 用POSTERR选项可以打印基于分类结果的分类准则的后验概率错误率估计。用NOPRINT选项可以取消结果的显示。

在DISCRIM过程中还可以使用PRIORS语句指定先验概率q_t的取法。“PRIORS EQUAL”指定等先验概率。“PRIORS PROPORTIONAL”指定先验概率与各类个数成正比。“PRIORS 概率值表”可以直接指定各组的先验概率值。

5.2.3 例子

用卫星遥感可以分辨作物的种类。CROPS是训练数据集, 其中包含了作物的实际种类(CROP)和四种遥感指标变量(X1-X4)。数据集中还把各X1-X4变量值作为一个字符型变量读入来作为行标识。

```
data crops;
   title '五种作物遥感数据的判别分析';
   input crop $ 1-10 x1-x4 xvalues $ 11-21;
   cards;
CORN      16 27 31 33
CORN      15 23 30 30
CORN      16 27 27 26
CORN      18 20 25 23
CORN      15 15 31 32
CORN      15 32 32 15
CORN      12 15 16 73
SOYBEANS  20 23 23 25
SOYBEANS  24 24 25 32
SOYBEANS  21 25 23 24
SOYBEANS  27 45 24 12
SOYBEANS  12 13 15 42
SOYBEANS  22 32 31 43
COTTON    31 32 33 34
COTTON    29 24 26 28
COTTON    34 32 28 45
COTTON    26 25 23 24
COTTON    53 48 75 26
COTTON    34 35 25 78
SUGARBEETS22 23 25 42
SUGARBEETS25 25 24 26
SUGARBEETS34 25 16 52
SUGARBEETS54 23 21 54
SUGARBEETS25 43 32 15
SUGARBEETS26 54  2 54
CLOVER    12 45 32 54
CLOVER    24 58 25 34
CLOVER    87 54 61 21
```

```
CLOVER    51 31 31 16
CLOVER    96 48 54 62
CLOVER    31 31 11 11
CLOVER    56 13 13 71
CLOVER    32 13 27 32
CLOVER    36 26 54 32
CLOVER    53 08 06 54
CLOVER    32 32 62 16
;
run;
```

用下列DISCRIM过程可以产生线性判别函数(METHOD=NORMAL规定使用参数方法, POOL=YES选项规定使用合并协方差阵, 这样产生的判别函数是线性函数)。用OUTSTAT=选项指定了判别函数的输出数据集为CROPSTAT, 这个数据集可以作为后续的DISCRIM过程输入用来判别检验数据集。选项LIST要求列出每个观测的结果, CROSSVALIDATE要求交叉核实。"PRIORS PROPORTIONAL"即按各种类出现的比例计算各类的先验概率, ID语句指定列出各观测时以什么变量值作为标识。

```
proc discrim data=crops outstat=cropstat
             method=normal pool=yes
             list crossvalidate;
   class crop;
   priors proportional;
   id xvalues;
   var x1-x4;
   title2 '使用线性判别函数';
run;
```

结果如下(节略):

```
                The DISCRIM Procedure

Observations    36     DF Total             35
Variables        4     DF Within Classes    31
Classes          5     DF Between Classes    4
```

上面是一些基本情况。

```
                  Class Level Information

             Variable                                          Prior
crop         Name         Frequency   Weight  Proportion  Probability

CLOVER       CLOVER              11  11.0000    0.305556     0.305556
CORN         CORN                 7   7.0000    0.194444     0.194444
COTTON       COTTON               6   6.0000    0.166667     0.166667
SOYBEANS     SOYBEANS             6   6.0000    0.166667     0.166667
SUGARBEETS   SUGARBEETS           6   6.0000    0.166667     0.166667
```

以上为各组的基本情况, 并列出了各组先验概率值。因为指定了“PRIORS PROPORTIONAL”所以各组的先验概率按实际数据中各组比例计算。

```
Pairwise Generalized Squared Distances Between Groups

   2          _   _         -1  _   _
  D (i|j) = (X - X )' COV    (X - X ) - 2 ln PRIOR
             i   j           i   j             j
```

上面为各组均值间广义距离平方的公式, 即$D_j^2(\bar{X}_i) = (\bar{X}_i - \bar{X}_j)'S^{-1}(\bar{X}_i - \bar{X}_j) - 2\ln q_j$。

```
                Linear Discriminant Function

                 _     -1 _                                  -1 _
Constant = -.5 X' COV    X  + ln PRIOR     Coefficient = COV    X
                j        j           j     Vector               j
```

上面即线性判别函数的公式, 给出了到第j类的线性判别函数的常数项和自变量的系数向量的公式。下面具体列出了各类的线性判别函数的各常数项及系数值。

Linear Discriminant Function for crop

Variable	CLOVER	CORN	COTTON	SOYBEANS	SUGARBEETS
Constant	-10.98457	-7.72070	-11.46537	-7.28260	-9.80179
x1	0.08907	-0.04180	0.02462	0.0000369	0.04245
x2	0.17379	0.11970	0.17596	0.15896	0.20988
x3	0.11899	0.16511	0.15880	0.10622	0.06540
x4	0.15637	0.16768	0.18362	0.14133	0.16408

比如, 观测了X1-X4后棉花类的线性判别函数为$-11.46537 + 0.02462x_1 + 0.017596x_2 + 0.15880x_3 + 0.18362x_4$。

下面为判别分析对训练数据集(Calibration Data)用线性判别函数的判别结果, 先给出了广义平方距离函数的公式和每个观测属于各类的后验概率的公式：

The DISCRIM Procedure
Classification Results for Calibration Data: WORK.CROPS
Resubstitution Results using Linear Discriminant Function

Generalized Squared Distance Function

$$D^2_j(X) = (X-\bar{X}_j)' \, COV^{-1} \, (X-\bar{X}_j) - 2 \ln PRIOR_j$$

```
        Posterior Probability of Membership in Each crop

                                  2                    2
         Pr(j|X) = exp(-.5 D (X)) / SUM exp(-.5 D (X))
                                  j       k            k
```

下面就是每个观测的判别情况, 包括原来为哪一类(From CROP), 分入了哪一类(Classified into CROP), 属于各类的后验概率值(Posterior Probability of Membership in CROP)。有星号的为错判的观测。

Posterior Probability of Membership in crop

xvalues	From crop	Classified into crop		CLOVER	CORN	COTTON	SOYBEANS	SUGARBEETS
16 27 31 33	CORN	CORN		0.0894	0.4054	0.1763	0.2392	0.0897
15 23 30 30	CORN	CORN		0.0769	0.4558	0.1421	0.2530	0.0722
16 27 27 26	CORN	CORN		0.0982	0.3422	0.1365	0.3073	0.1157
18 20 25 23	CORN	CORN		0.1052	0.3634	0.1078	0.3281	0.0955
15 15 31 32	CORN	CORN		0.0588	0.5754	0.1173	0.2087	0.0398
15 32 32 15	CORN	SOYBEANS	*	0.0972	0.3278	0.1318	0.3420	0.1011
12 15 16 73	CORN	CORN		0.0454	0.5238	0.1849	0.1376	0.1083
20 23 23 25	SOYBEANS	SOYBEANS		0.1330	0.2804	0.1176	0.3305	0.1385
24 24 25 32	SOYBEANS	SOYBEANS		0.1768	0.2483	0.1586	0.2660	0.1502
21 25 23 24	SOYBEANS	SOYBEANS		0.1481	0.2431	0.1200	0.3318	0.1570
27 45 24 12	SOYBEANS	SUGARBEETS	*	0.2357	0.0547	0.1016	0.2721	0.3359
12 13 15 42	SOYBEANS	CORN	*	0.0549	0.4749	0.0920	0.2768	0.1013
22 32 31 43	SOYBEANS	COTTON	*	0.1474	0.2606	0.2624	0.1848	0.1448
31 32 33 34	COTTON	CLOVER	*	0.2815	0.1518	0.2377	0.1767	0.1523
29 24 26 28	COTTON	SOYBEANS	*	0.2521	0.1842	0.1529	0.2549	0.1559
34 32 28 45	COTTON	CLOVER	*	0.3125	0.1023	0.2404	0.1357	0.2091
26 25 23 24	COTTON	SOYBEANS	*	0.2121	0.1809	0.1245	0.3045	0.1780
53 48 75 26	COTTON	CLOVER	*	0.4837	0.0391	0.4384	0.0223	0.0166
34 35 25 78	COTTON	COTTON		0.2256	0.0794	0.3810	0.0592	0.2548
22 23 25 42	SUGARBEETS	CORN	*	0.1421	0.3066	0.1901	0.2231	0.1381
25 25 24 26	SUGARBEETS	SOYBEANS	*	0.1969	0.2050	0.1354	0.2960	0.1667
34 25 16 52	SUGARBEETS	SUGARBEETS		0.2928	0.0871	0.1665	0.1479	0.3056
54 23 21 54	SUGARBEETS	CLOVER	*	0.6215	0.0194	0.1250	0.0496	0.1845
25 43 32 15	SUGARBEETS	SOYBEANS	*	0.2258	0.1135	0.1646	0.2770	0.2191
26 54 2 54	SUGARBEETS	SUGARBEETS		0.0850	0.0081	0.0521	0.0661	0.7887
12 45 32 54	CLOVER	COTTON	*	0.0693	0.2663	0.3394	0.1460	0.1789
24 58 25 34	CLOVER	SUGARBEETS	*	0.1647	0.0376	0.1680	0.1452	0.4845
87 54 61 21	CLOVER	CLOVER		0.9328	0.0003	0.0478	0.0025	0.0165
51 31 31 16	CLOVER	CLOVER		0.6642	0.0205	0.0872	0.0959	0.1322
96 48 54 62	CLOVER	CLOVER		0.9215	0.0002	0.0604	0.0007	0.0173
31 31 11 11	CLOVER	SUGARBEETS	*	0.2525	0.0402	0.0473	0.3012	0.3588
56 13 13 71	CLOVER	CLOVER		0.6132	0.0212	0.1226	0.0408	0.2023
32 13 27 32	CLOVER	CLOVER		0.2669	0.2616	0.1512	0.2260	0.0943
36 26 54 32	CLOVER	COTTON	*	0.2650	0.2645	0.3495	0.0918	0.0292
53 08 06 54	CLOVER	CLOVER		0.5914	0.0237	0.0676	0.0781	0.2392
32 32 62 16	CLOVER	COTTON	*	0.2163	0.3180	0.3327	0.1125	0.0206

```
* Misclassified observation
```

下面给出了训练数据判别的概况, 先写出了广义平方距离的公式和属于每一类的后验概率的公式(略), 然后是每一类判入各类的个数和百分比：

```
   Number of Observations and Percent Classified into crop

From crop   CLOVER    CORN  COTTON SOYBEANS SUGARBEETS   Total

CLOVER          6       0       3        0          2      11
            54.55    0.00   27.27     0.00      18.18  100.00

CORN            0       6       0        1          0       7
             0.00   85.71    0.00    14.29       0.00  100.00

COTTON          3       0       1        2          0       6
            50.00    0.00   16.67    33.33       0.00  100.00

SOYBEANS        0       1       1        3          1       6
             0.00   16.67   16.67    50.00      16.67  100.00

SUGARBEETS      1       1       0        2          2       6
            16.67   16.67    0.00    33.33      33.33  100.00

Total          10       8       5        8          5      36
            27.78   22.22   13.89    22.22      13.89  100.00

Priors     0.30556 0.19444 0.16667  0.16667    0.16667
```

比如, CLOVER(苜蓿)一共有11个观测, 正确判别的为6个, 占54.55, 有3个错判为COTTON(棉花), 2个错判为SUGARBEETS(甜菜)。最后一行为各类的先验概率。下面为各类的错判率(把某类错判为其它类的次数百分比)：

```
            Error Count Estimates for crop

           CLOVER    CORN  COTTON  SOYBEANS  SUGARBEETS   Total
```

```
Rate        0.4545  0.1429  0.8333    0.5000      0.6667  0.5000
Priors      0.3056  0.1944  0.1667    0.1667      0.1667
```

可见识别最好的是玉米, 最差的是棉花。

下面是对训练数据集进行交叉核实判别的情况。交叉核实的想法是, 为了判断观测i的判别正确与否, 用删除第i个观测的训练数据集算出判别规则(函数), 然后用此判别函数来判别第i观测。对每一观测都进行这样的判别。结果先写出了广义平方距离函数和后验概率公式。这里因为建立判别规则时不使用要判别的观测, 所以公式中用了$\bar{X}_{(X)j}$表示除去了X所在观测后的第j组的均值, 用$\mathrm{COV}_{(X)}$表示除去X所在观测后得到的合并协方差阵估计。

```
                    The DISCRIM Procedure
   Classification Summary for Calibration Data: WORK.CROPS
 Cross-validation Summary using Linear Discriminant Function

           Generalized Squared Distance Function

       2          _           -1     _
      D (X) = (X-X    )' COV    (X-X    ) - 2 ln PRIOR
       j          (X)j      (X)     (X)j             j

       Posterior Probability of Membership in Each crop

                          2                    2
        Pr(j|X) = exp(-.5 D (X)) / SUM exp(-.5 D (X))
                          j        k           k
```

后面是对各类交叉核实判别的概况。

```
   Number of Observations and Percent Classified into crop

From crop   CLOVER    CORN  COTTON SOYBEANS SUGARBEETS   Total

CLOVER           4       3       1        0          3      11
```

```
                36.36   27.27    9.09     0.00      27.27  100.00

CORN               0       4       1        2          0       7
                0.00   57.14   14.29    28.57       0.00  100.00

COTTON             3       0       0        2          1       6
               50.00    0.00    0.00    33.33      16.67  100.00

SOYBEANS           0       1       1        3          1       6
                0.00   16.67   16.67    50.00      16.67  100.00

SUGARBEETS         2       1       0        2          1       6
               33.33   16.67    0.00    33.33      16.67  100.00

Total              9       9       3        9          6      36
               25.00   25.00    8.33    25.00      16.67  100.00

Priors      0.30556 0.19444 0.16667  0.16667    0.16667
```

用交叉核实评价现在11个苜蓿的观测只判对了4个。下面是用交叉核实计算的各类的错判率：

```
          Error Count Estimates for crop

         CLOVER     CORN   COTTON  SOYBEANS  SUGARBEETS   Total

Rate     0.6364   0.4286   1.0000    0.5000      0.8333  0.6667
Priors   0.3056   0.1944   0.1667    0.1667      0.1667
```

这时错误最少的玉米也有43%的错判率。

现在假设我们有若干遥感数据放在了数据集TEST中, 实际是已知作物类型的(在变量CROP中), 但是我们假装不知道然后用上面建立的线性判别函数(已保存在CROPSTAT数据集中)对这些遥感数据进行判别, 这样可以得到比较客观的判别效果的评价。下面程序中用DATA=指定了判别函数数据集(由上一次的DISCRIM过程产生), 用TESTDATA=选项指定了检验数据集名, 用TESTOUT=选项指定了检验数据集判别结

果的输出数据集, 用TESTLIST要求列出检验结果。TESTID语句指定检验数据集的各观测用什么变量的值来标识。

```
data test;
   input crop $ 1-10 x1-x4 xvalues $ 11-21;
   cards;
CORN      16 27 31 33
SOYBEANS  21 25 23 24
COTTON    29 24 26 28
SUGARBEETS54 23 21 54
CLOVER    32 32 62 16
;

proc discrim data=cropstat
     testdata=test testout=tout testlist;
   class crop;
   testclass crop;
   testid xvalues;
   var x1-x4;
   title2 '检验数据的判别';
run;

proc print data=tout;
   title2 '检验数据的判别结果';
run;
```

结果列出了每个观测的判别结果和判入每类的后验概率, 因为我们知道真实类, 所以结果中有一项是“From CROP”, 如果不知道真实类则只能给出判入的类(Classified into CROPP)。

```
                Posterior Probability of Membership in crop

                      Classified
xvalues     From crop into crop     CLOVER    CORN   COTTON SOYBEANS SUGARBEETS

16 27 31 33 CORN      CORN          0.0894  0.4054   0.1763   0.2392     0.0897
21 25 23 24 SOYBEANS  SOYBEANS      0.1481  0.2431   0.1200   0.3318     0.1570
29 24 26 28 COTTON    SOYBEANS   *  0.2521  0.1842   0.1529   0.2549     0.1559
```

```
54 23 21 54 SUGARBEETS CLOVER     *  0.6215  0.0194   0.1250   0.0496     0.1845
32 32 62 16 CLOVER     COTTON     *  0.2163  0.3180   0.3327   0.1125     0.0206

                  * Misclassified observation
```

下面给出了各类的判别概况（略）, 错判的百分比。

```
                Error Count Estimates for crop

           CLOVER    CORN  COTTON  SOYBEANS  SUGARBEETS   Total

Rate       1.0000  0.0000  1.0000    0.0000      1.0000  0.6389
Priors     0.3056  0.1944  0.1667    0.1667      0.1667
```

可见错判率很高(总错判率达64%)。

5.3 聚类分析

聚类分析和判别分析有相似的作用, 都是起到分类的作用。但是, 判别分析是已知分类然后总结出判别规则, 是一种有指导的学习；而聚类分析则是有了一批样本, 不知道它们的分类, 甚至连分成几类也不知道, 希望用某种方法把观测进行合理的分类, 使得同一类的观测比较接近, 不同类的观测相差较多, 这是无指导的学习。

所以, 聚类分析依赖于对观测间的接近程度(距离)或相似程度的理解, 定义不同的距离量度和相似性量度就可以产生不同的聚类结果。

SAS/STAT中提供了谱系聚类、快速聚类、变量聚类等聚类过程。

5.3.1 谱系聚类方法介绍

谱系聚类是一种逐次合并类的方法, 最后得到一个聚

类的二叉树聚类图。其想法是, 对于n个观测, 先计算其两两的距离得到一个距离矩阵, 然后把离得最近的两个观测合并为一类, 于是我们现在只剩了$n-1$个类(每个单独的未合并的观测作为一个类)。计算这$n-1$个类两两之间的距离, 找到离得最近的两个类将其合并, 就只剩下了$n-2$个类, ..., 直到剩下两个类, 把它们合并为一个类为止。当然, 真的合并成一个类就失去了聚类的意义, 所以上面的聚类过程应该在某个类水平数(即未合并的类数)停下来, 最终的类就取这些未合并的类。决定聚类个数是一个很复杂的问题。

设观测个数为n, 变量个数为ν, G为在某一聚类水平上的类的个数, x_i为第i个观测, C_K是当前(水平G)的第K类, N_K为C_K中的观测个数, $\bar{X}$为均值向量, $\bar{X}_K$为类C_K中的均值向量(中心), $||x||$为欧氏长度, $T = \Sigma_{i=1}^{n} \|x_i - \bar{X}\|^2$为总离差平方和, $W_K = \Sigma_{i \in C_K} \|x_i - \bar{X}_K\|^2$为类$C_K$的类内离差平方和, $P_G = \Sigma W_J$为聚类水平G对应的各类的类内离差平方和的总和。假设某一步聚类把类C_K和类C_L合并为下一水平的类C_M, 则定义$B_{KL} = W_M - W_K - W_L$为合并导致的类内离差平方和的增量。用$d(x,y)$代表两个观测之间的距离或非相似性测度, D_{KL}为第G水平的类C_K和类C_L之间的距离或非相似性测度。

进行谱系聚类时, 类间距离可以直接计算, 也可以从上一聚类水平的距离递推得到。观测间的距离可以用欧氏距离或欧氏距离的平方, 如果用其它距离或非相似性测度得到了一个观测间的距离矩阵也可以作为谱系聚类方法的输入。

根据类间距离的计算方法的不同, 有多种不同的聚类方法。其中几种介绍如下：

一、类平均法(METHOD=AVERAGE)

测量两类每对观测间的平均距离, 即

$$D_{KL} = \frac{1}{N_K N_L} \sum_{i \in C_K} \sum_{j \in C_L} d(x_i, x_j)$$

在$d(x,y) = \|x-y\|^2$时若类C_K和类C_L合并为下一水平的类C_M, 则合并得到的类C_M和其他的类C_J之间距离的递推公式为

$$D_{JM} = (N_K D_{JK} + N_L D_{JL})/N_M$$

二、重心法(METHOD=CENTROID)

重心法测量两个类的重心(均值)之间的(平方)欧氏距离。即

$$D_{KL} = \|\bar{X}_K - \bar{X}_L\|^2$$

当观测间距离为平方欧氏距离时有递推公式

$$D_{JM} = (N_K D_{JK} + N_L D_{JL})/N_M - N_K N_L D_{KL}/N_M^2$$

三、最长距离法(METHOD=COMPLETE)

计算两类观测间最远一对的距离, 即

$$D_{KL} = \max_{i \in C_K} \max_{j \in C_L} d(x_i, x_j)$$

递推公式为

$$D_{JM} = \max(D_{JK}, D_{JL})$$

四、最短距离法(METHOD=SINGLE)

计算两类观测间最近一对的距离, 即

$$D_{KL} = \min_{i \in C_K} \min_{j \in C_L} d(x_i, x_j)$$

递推公式为

$$D_{JM} = \min(D_{JK}, D_{JL})$$

五、密度估计法(METHOD=DENSITY)

密度估计法按非参数密度来定义两点间的距离d^*。如果两个点x_i和x_j是近邻(两点距离小于某指定常数或x_j在距离x_i最近的若干点内) 则距离是两点密度估计的倒数的平均, 否则距离为正无穷。密度估计有最近邻估计(K=)、均匀核估计(R=)和Wong混合法(HYBRID)。

六、Ward最小方差法(METHOD=WARD)

也称Ward离差平方和法。组间距离为

$$D_{KL} = B_{KL} = \|\bar{X}_K - \bar{X}_L\|^2/(1/N_K + 1/N_L)$$

当观测间距离为$d(x, y) = \|x - y\|^2/2$时递推公式为

$$D_{JM} = \left((N_J + N_K)D_{JK} + (N_J + N_L)D_{JL} - N_J D_{KL}\right)/(N_J + N_M)$$

Ward方法并类时总是使得并类导致的类内离差平方和增量最小。

其它的聚类方法还有EML 法、可变类平均法(FLEXIBLE)、McQuitty 相似分析法(MCQUITTY)、中间距离法(MEDIAN)、两阶段密度估计法(TWOSTAGE)等。

5.3.2 谱系聚类类数的确定

谱系聚类最终得到一个聚类树, 可以把所有观测聚为一类。到底应该把观测分为几类是一个比较困难的问题, 因为分类问题本身就是没有一定标准的, 关于这一点《实用多元统计分析》(王学仁、王松桂, 上海科技出版社)第十章给出了一个很好的例子, 即扑克牌的分类。我们可以把扑克牌按花色分类, 按大小点分类, 按桥牌的高花色低花色分类, 等等。

决定类数的一些方法来自统计的方差分析的思想, 我们在这里作一些介绍。

一、R^2统计量

$$R^2 = 1 - \frac{P_G}{T}$$

其中P_G为分类数为G个类时的总类内离差平方和, T为所有变量的总离差平方和。R^2越大, 说明分为G个类时每个类内的离差平方和都比较小, 也就是分为G个类是合适的。但是, 显然分类越多, 每个类越小, R^2越大, 所以我们只能取G使得R^2足够大, 但G本身比较小, 而且R^2不再大幅度增加。

二、半偏相关

在把类C_K和类C_L合并为下一水平的类C_M时, 定义半偏相关

$$\text{半偏}R^2 = \frac{B_{KL}}{T}$$

其中B_{KL}为合并类引起的类内离差平方和的增量, 半偏相关越大, 说明这两个类越不应该合并, 所以如果由G类合并为$G-1$类时如果半偏相关很大就应该取G类。

三、双峰性系数

$$b = (m_3^2 + 1)/(m_4 + 3(n-1)^2/((n-2)(n-3)))$$

其中m_3是偏度, m_4是峰度。大于0.555的b值(这时为均匀分布)指示可能有双峰或多峰边缘分布。最大值1.0(二值分布)从仅取两值的总体得到。

四、伪F统计量

$$F = \frac{(T - P_G)/(G-1)}{P_G/(n-G)}$$

伪F统计量评价分为G个类的效果。如果分为G个类合理, 则类内离差平方和(分母)应该较小, 类间平方和(分子)相对较大。所以应该取伪F统计量较大而类数较小的聚类水平。

五、伪t^2统计量

$$t^2 = B_{KL}/\left((W_K + W_L)/(N_K + N_L - 2)\right)$$

用此统计量评价合并类C_K和类C_L的效果, 该值大说明不应合并这两个类, 所以应该取合并前的水平。

5.3.3 用CLUSTER过程和TREE过程进行谱系聚类

一、CLUSTER过程用法

CLUSTER过程的一般格式为：

```
PROC CLUSTER DATA=输入数据集
                METHOD=聚类方法 选项;
    VAR 聚类用变量;
    COPY 复制变量;
RUN;
```

其中的VAR语句指定用来聚类的变量。COPY语句把指定的变量复制到OUTTREE＝的数据集中。

PROC CLUSTER语句的主要选项有：

- METHOD=选项, 这是必须指定的, 此选项决定我们要用的聚类方法, 主要由类间距离定义决定。方法有AVERAGE, CENTROID, COMPLETE, SINGLE, DENSITY, WARD, EML, FLEXIBLE, MCQUITTY, MEDIAN, TWOSTAGE等, 其中DENSITY, TWOSTAGE等方法还要额外指定密度估计方法(K=, R=或HYBRID)。
- 输入DATA＝数据集, 可以是原始观测数据集, 也可以是距离矩阵数据集。
- OUTTREE=输出谱系聚类树数据集, 把谱系聚类树输出到一个数据集, 可以用TREE过程绘图并实际分类。
- STANDARD选项, 把变量标准化为均值0, 标准差1。

- PSEUDO选项和CCC选项。PSEUDO选项要求计算伪F和伪t^2统计量, CCC选项要求计算R^2、半偏R^2和CCC统计量。其中CCC统计量也是一种考察聚类效果的统计量, CCC较大的聚类水平是较好的。

二、TREE过程用法

TREE过程可以把CLUSTER过程产生的OUTTREE＝数据集作为输入, 画出谱系聚类的树图, 并按照用户指定的聚类水平(类数)产生分类结果数据集。一般格式如下：

```
PROC TREE DATA＝输入聚类结果数据集
          OUT=输出数据集 GRAPHICS
          NCLUSTER=类数 选项;
    COPY 复制变量;
RUN;
```

其中COPY语句把输入数据集中的变量复制到输出数据集(实际上这些变量也必须在CLUSTER过程中用COPY语句复制到OUTTREE＝数据集)。PROC TREE语句的重要选项有：

- DATA＝数据集, 指定从CLUSTER过程生成的OUTTREE＝数据集作为输入。
- OUT＝数据集, 指定包含最后分类结果(每一个观测属于哪一类, 用一个CLUSTER变量区分)的输出数据集。

- NCLUSTERS＝选项, 由用户指定最后把样本观测分为多少个类。
- HORIZONTAL, 画树图时横向画。

三、例子

我们以多元分析中一个经典的数据作为例子, 这是Fisher分析过的鸢尾花数据, 有三种不同鸢尾花(Setosa、Versicolor、Virginica), 种类信息存入了变量SPECIES, 并对每一种测量了50棵植株的花瓣长(PETALLEN), 花瓣宽(PETALWID), 花萼长(SEPALLEN), 花萼宽(SEPALWID)。这个数据已知分类, 并不属于聚类分析的研究范围。这里我们为了示例, 假装不知道样本的分类情况(既不知道类数也不知道每一个观测属于的类别), 让SAS取进行聚类分析, 如果得到的类数和分类结果符合真实的植物分类, 我们就可以知道聚类分析产生了好的结果。

这里我们假定数据已输入SASUSER.IRIS中(见系统帮助菜单的"SAS System Help - Sample Programs and Applications - SAS/STAT - Documentation Example 3 from Proc Cluster")。为了进行谱系聚类并产生帮助确定类数的统计量, 使用如下过程:

```
proc cluster data=sasuser.iris method=ward
             outtree=otree pseudo ccc;
  var petallen petalwid sepallen sepalwid;
  copy species;
run;
```

可以显示如下的聚类过程(节略):

Cluster History

NCL	-Clusters	Joined--	FREQ	SPRSQ	RSQ	ERSQ	CCC	PSF	PST2	Tie
149	OB16	OB76	2	0.0000	1.00	.	.	.	.	
148	OB2	OB58	2	0.0000	1.00	.	.	1854	.	T
147	OB96	OB107	2	0.0000	1.00	.	.	1400	.	T
146	OB89	OB113	2	0.0000	1.00	.	.	1253	.	T
145	OB65	OB126	2	0.0000	1.00	.	.	1183	.	T
.	.	.	.	.	.	.	.	.	.	.
25	CL50	OB57	7	0.0006	.982	.973	6.45	291	5.6	
24	CL78	CL62	7	0.0007	.982	.972	6.43	294	9.8	
23	CL68	CL38	9	0.0008	.981	.971	6.40	296	6.9	
22	CL30	OB137	6	0.0009	.980	.970	6.35	298	5.1	
21	CL70	CL33	4	0.0010	.979	.969	6.29	301	3.2	
20	CL36	OB25	10	0.0011	.978	.967	6.21	303	9.8	
19	CL40	CL22	19	0.0011	.977	.966	6.15	306	7.7	
18	CL25	CL39	10	0.0012	.976	.964	6.08	309	6.2	
17	CL29	CL45	16	0.0014	.974	.962	6.03	314	8.2	
16	CL34	CL32	15	0.0015	.973	.960	5.98	318	9.0	
15	CL24	CL28	15	0.0016	.971	.958	5.93	324	9.8	
14	CL21	CL53	7	0.0019	.969	.955	5.85	329	5.1	
13	CL18	CL48	15	0.0023	.967	.953	5.69	334	8.9	
12	CL16	CL23	24	0.0023	.965	.950	4.63	342	9.6	
11	CL14	CL43	12	0.0025	.962	.946	4.67	353	5.8	
10	CL26	CL20	22	0.0027	.959	.942	4.81	368	12.9	
9	CL27	CL17	31	0.0031	.956	.936	5.02	387	17.8	
8	CL35	CL15	23	0.0031	.953	.930	5.44	414	13.8	
7	CL10	CL47	26	0.0058	.947	.921	5.43	430	19.1	
6	CL8	CL13	38	0.0060	.941	.911	5.81	463	16.3	
5	CL9	CL19	50	0.0105	.931	.895	5.82	488	43.2	
4	CL12	CL11	36	0.0172	.914	.872	3.99	515	41.0	
3	CL6	CL7	64	0.0301	.884	.827	4.33	558	57.2	
2	CL4	CL3	100	0.1110	.773	.697	3.83	503	116	
1	CL5	CL2	150	0.7726	.000	.000	0.00	.	503	

这个输出列出了把150个观测每次合并两类, 共合并149次的过程。NCL列指定了聚类水平G(即这一步存在的单独的类数)。“-Clusters Joined-”为两列, 指明这一步合并了哪两个类。其中OB*xxx*表示哪一个原始观测, 而CL*xxx*表示在哪一个聚类水平上产生的类。比如, NCL为149时合并的是OB16和OB76, 即16号观测和76号观测, NCL为1(最后一次合并)合并的是CL5和CL2, 即类水平为5时得到的类和类水平为2时得到的类, CL5又是由CL9和CL19合并得到的, CL2是由CL4和CL3合并得到的, 等等。FREQ表示这

次合并得到的类有多少个观测。SPRSQ是半偏R^2, RSQ是R^2, ERSQ是在均匀零假设下的R^2的近似期望值, CCC为CCC统计量, PSF为伪F统计量, PST2为伪t^2统计量, Tie指示距离最小的候选类对是否有多对。

因为我们假装不知道数据的实际分类情况, 所以我们必须找到一个合理的分类个数。为此, 考察CCC、伪F、伪t^2和半偏R^2统计量。我们打开INSIGHT界面, 调入上面产生的OTREE数据集, 绘制各统计量的图形。因为类水平太大时的信息没有多少用处, 所以我们对OTREE数据集取其类水平不超过30的观测, 即:

```
data ot;
  set otree;
  where _ncl_ <= 30;
run;
```

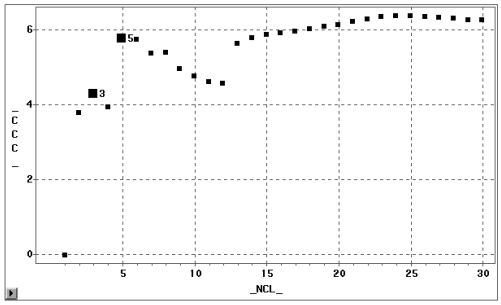


图 5.2: 聚类分析: CCC统计量

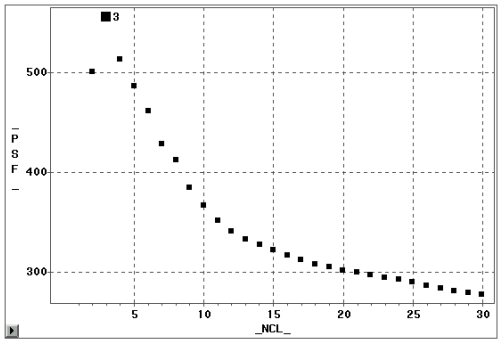


图 5.3: 聚类分析: 伪F统计量

各统计量的图形见图5.2—图5.5。CCC统计量建议取5类或3类(局部最大值), 伪F建议3类(局部最大值), 伪t^2建议2或3类(局部最大值处是不应合并的, 即局部

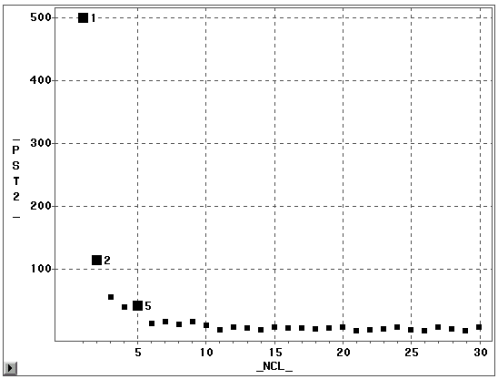


图 5.4: 聚类分析: 伪t^2统计量

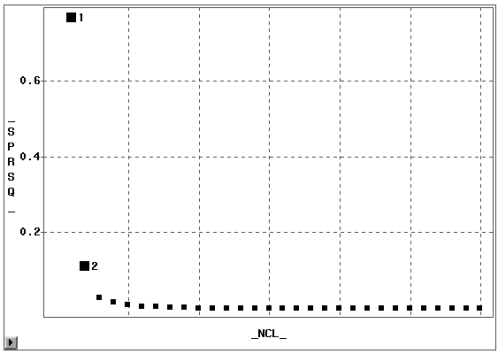


图 5.5: 聚类分析: 半偏R^2

最大值处的类数加1), 半偏R^2建议2或3类。由这些指标看比较一致的是3类, 其次是2类和5类。为了看为什么不能明显地分为三类, 我们对四个变量求主分量, 画出前两个主分量的散点图(见图5.6)。可以看出Setosa(方块)与其它两类分得很开, 而Versicolor(正三角)与Virginica(倒三角)则不易分开。

因为我们知道要分成3类, 所以我们用如下的TREE过程绘制树图并产生分类结果数据集:

```
proc tree data=otree graphics
     horizontal nclusters=3 out=oclust;
  copy species;
run;
```

树图见图5.7, 因为观测过多所以图显得杂乱。从图中也可以看出, 分为两类可以分得很开, 而分成三类时距离则不够远。如果上面的TREE过程去掉输出数据集要求, 可以用包含最后的聚类过程的OT数据集来作为输入, 见图5.8。这个TREE过程用NCLUSTERS=3指定了分成3个类, 结果数据

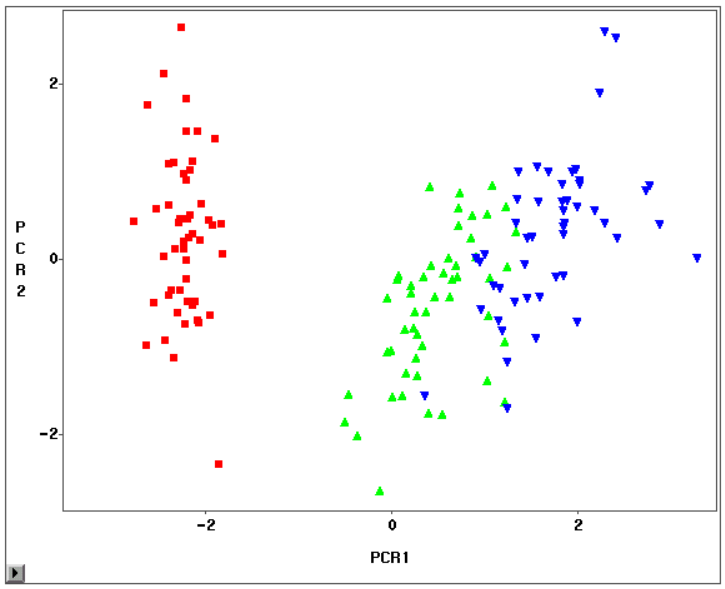


图 5.6: 聚类分析: IRIS数据的前两个主分量

集OCLUST中有一个CLUSTER变量代表生成的分类。

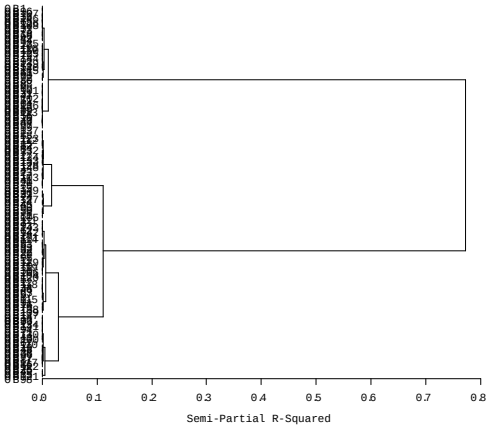


图 5.7: 聚类分析: 聚类树

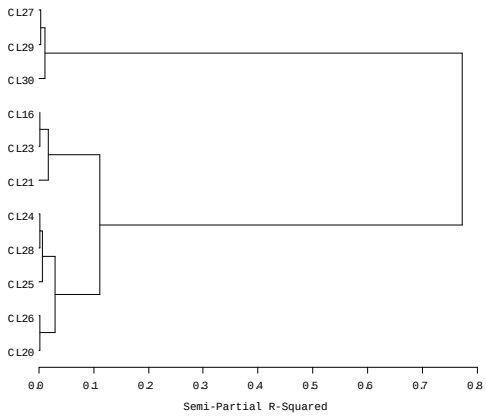


图 5.8: 聚类分析: 简化的聚类树

为了统计分类结果, 可以用FREQ过程作表：

```
proc freq data=oclust;
  tables species*cluster /
      nopct norow nocol;
run;
```

得

```
            Table of species by CLUSTER

    species(Species)     CLUSTER

    Frequency  |       1|       2|       3|  Total
    -----------+--------+--------+--------+
    Setosa     |       0 |      0 |     50 |     50
    -----------+--------+--------+--------+
    Versicolor |     49 |       1 |      0 |     50
    -----------+--------+--------+--------+
    Virginica  |     15 |     35 |      0 |     50
    -----------+--------+--------+--------+
    Total             64       36       50      150
```

可见Virginica被分错的较多。

读者可以自己试用其它的类间距离来聚类, 可以得到不同的结果。

练习

1. 对SASUSER.GPA中的变量进行主分量分析并试图解释结果。
2. 对SASUSER.GPA中的变量进行因子分析并试进行旋转, 对得到的因子进行解释。
3. 设有三个组, 四个变量, 数据见表5.1。计算线性判别函数, 简述对训练样本的判别情况。

表 5.1: 练习3的数据

组别	X1	X2	X3	X4
1	6	-11.5	19	90
1	-4	-15.0	13	54
1	0	-23.0	5	-35
1	-100	-21.4	7	-15
1	-5	-18.5	15	18
1	10	-18.0	14	50
1	-8	-14.0	16	56
2	90.2	-17.0	17	3
2	0	-14.0	20	35
2	-100	-21.5	15	-40
2	13	-17.2	18	2
3	-11	-18.5	25	-36
3	0.5	-11.5	19	37
3	-10	-19.0	21	-42
3	20	-22.0	8	-20
3	0.6	-13.0	26	21
3	-40	-20.0	22	-50

4. 对表5.1中的例子数据集SOCECON作聚类分析。

第六章 S语言介绍

6.1 S快速入门

6.1.1 背景介绍

S语言是由AT&T贝尔实验室开发的一种用来进行数据探索、统计分析、作图的解释型语言。它的丰富的数据类型(向量、数组、列表、对象等)特别有利于实现新的统计算法, 其交互式运行方式及强大的图形及交互图形功能使得我们可以方便地探索数据。

目前S语言的实现版本主要是S-PLUS。它基于S语言, 并由MathSoft公司的统计科学部进一步完善。作为统计学家及一般研究人员的通用方法工具箱, S-PLUS强调演示图形、探索性数据分析、统计方法、开发新统计工具的计算方法, 以及可扩展性。

S-PLUS可以直接用来进行标准的统计分析得到所需结果, 但是它的主要的特点是它可以交互地从各个方面去发现数据中的信息, 并可以很容易地实现一个新的统计方法。

S-PLUS有微机版本和工作站版本, 它是一个商业软件。Auckland大学的Robert Gentleman 和Ross Ihaka 及其他志愿人员开发了一个R系统, 其语法形式与S语言基本相同, 但实现不同, 两种语言的程

序有一定的兼容性。R是一个GPL自由软件, 现在的版本是1.4.1版, 它比S-PLUS还少许多功能, 但已经具有了很强的实用性。我们在这里尽量介绍S-PLUS和R都能使用的功能, 且以R为主。下面我们用S统称S-PLUS和R。

6.1.2 入门实例

S的基本界面是一个交互式命令窗口, 命令提示符是一个大于号, 命令的结果马上显示在命令下面。S命令主要有两种形式：表达式或赋值运算(用<-表示)。在命令提示符后键入一个表达式表示计算此表达式并显示结果。赋值运算把赋值号右边的值计算出来赋给左边的变量。可以用向上光标键来找回以前运行的命令再次运行或修改后再运行。

S是区分大小写的, 所以x和X是不同的名字。

我们用一些例子来看S-PLUS的特点。假设我们已经进入了S-PLUS(或R)的交互式窗口。输入以下语句(大于号是系统提示,不是自己输入的)：

```
> x1 <- 0:100
> x2 <- x1*2*pi/100
> y <- sin(x2)
> plot(x,y, type='l')
```

这些语句可以绘制正弦曲线图。其中, "<-"是赋值运算符。0:100表示一个从0到100的等差数列向量。从第二个语句可以看出, 我们可以对向量直接进行四则运算, 计算得到的x2是向量x1的所有元素乘以常数2*pi/100的结果。从第三个语句可以看到函数可以

以向量为输入, 并可以输出一个向量, 结果向量y的每一个分量是自变量x2的每一个分量的正弦函数值。从最后一个语句可以看出函数的调用也很自由, 可以按位置给出自变量, 也可以用“自变量名=”的形式指定自变量值, 这样可以使用缺省值。

下面我们看一看S的统计功能。

```
> marks <- c(10, 6, 4, 7, 8)
> mean(marks)
[1] 7
> sd(marks)
[1] 2.236068
> median(marks)
[1] 7
> min(marks)
[1] 4
> max(marks)
[1] 10
> boxplot(marks)
>
```

第一个语句输入若干数据到一个向量, 函数c()用来把数据组合为一个向量。后面用了几个函数来计算数据的均值、标准差、中位数、最小值、最大值。最后的函数绘制数据的盒形图。例中sd()是R中才有的函数, 在S-PLUS中要用sqrt(var())来计算。在S命令方式中要显示一个表达式的值只要键入它。

为了演示S的回归计算, 我们把SAS中的例子数据SASUSER.CLASS 输出到了一个文本文件CLASS.TXT, SAS程序如下：

```
data _null_;
  set sasuser.class;
  file 'class.txt';
```

```
  put name sex age height weight;
run;
```

我们在R中把这个文件读入为为一个数据框(data frame, 相当于SAS中的数据集), 并进行回归, 绘制数据散点图和回归直线。程序及结果如下, 注意第二行的加号是系统给出的续行提示。

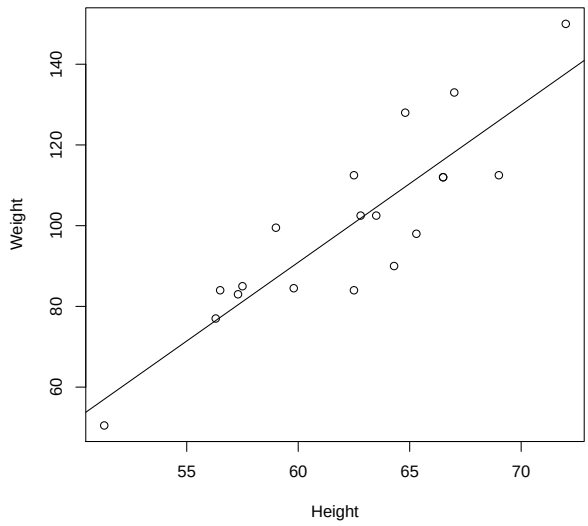


图 6.1: 用R产生的回归图形

```
> cl <- read.table("c:/work/class.txt", as.is=c(1),
+ col.names=c("Name", "Sex", "Age", "Height", "Weight"))
> cl
      Name Sex Age Height Weight
1    Alice   F  13   56.5   84.0
2    Becka   F  13   65.3   98.0
3     Gail   F  14   64.3   90.0
4    Karen   F  12   56.3   77.0
5    Kathy   F  12   59.8   84.5
6     Mary   F  15   66.5  112.0
7    Sandy   F  11   51.3   50.5
```

```
8   Sharon   F  15   62.5  112.5
9    Tammy   F  14   62.8  102.5
10  Alfred   M  14   69.0  112.5
11    Duke   M  14   63.5  102.5
12   Guido   M  15   67.0  133.0
13   James   M  12   57.3   83.0
14 Jeffrey   M  13   62.5   84.0
15    John   M  12   59.0   99.5
16  Philip   M  16   72.0  150.0
17  Robert   M  12   64.8  128.0
18  Thomas   M  11   57.5   85.0
19 William   M  15   66.5  112.0

> attach(cl)
> plot(Height, Weight)
> fit1 <- lm(Weight ~ Height)
> fit1

Call:
lm(formula = Weight ~ Height)

Coefficients:
(Intercept)       Height
   -143.027        3.899

> abline(fit1)
>
```

结果可以产生图6.1。我们从这个例子可以看到, 分析由函数完成, 结果也是对象, 可以作为进一步分析的输入。

S-PLUS也提供了一般的计算功能。比如, 求一个矩阵的逆:

```
> A <- matrix(c(1,2, 7,3), ncol=2, byrow=T)
> A
     [,1]  [,2]
[1,]    1     2
```

```
[2,]    7    3
> Ai <- solve(A)
> Ai
           [,1]       [,2]
[1,] -0.2727273  0.18181818
[2,]  0.6363636 -0.09090909
```

还可以进行矩阵运算, 如：

```
> b <- c(2,3)
> x <- Ai %*% b
> x
              [,1]
[1,] -2.220446e-16
[2,]  1.000000e+00
```

这实际上是解了一个线性方程组。“%*%”表示矩阵乘法。

可以把若干行命令保存在一个文本文件(比如C:\WORK\MYPROG.R)中, 然后用source函数来运行整个文件：

```
> source("C:\\WORK\\MYPROG.R")
```

注意字符串中的反斜杠要写成两个。用sink()函数可以把以后的输出从屏幕窗口转向到一个外部文本文件, 例如：

```
> sink("C:\\WORK\\MYPROG.OUT")
```

要恢复输出到屏幕窗口, 使用：

```
> sink()
```

在S中可以用“?”号后面跟要查询的函数来显示帮助信息, 例如

```
> ?c
> ?"=="
```

(查询特殊符号要用撇号括起来)。另外, 在MS Windows版的S-PLUS中可以用Windows帮助来得到帮助, 而在R中可以用HTML浏览器来查询帮助(用“help.start()函数或Help - R language (html)”启动)。用好S语言重要的一点就是要用熟系统的帮助功能, 因为S的教材或手册都不能列出所有的细节, 如果列出所有细节则过于繁琐了。

要退出S-PLUS或R, 可以用q()函数, 也可以用菜单命令。R在退出时提问是否保存当前工作空间, 它可以把当前定义的所有对象(有名字的向量、矩阵、列表、函数等)保存到一个文件。S-PLUS可以选择保存那些对象。

6.2 S向量

S语言是基于对象的语言, 不过它的最基本的数据还是一些预先定义好的数据类型, 如向量、矩阵、列表(list)等。更复杂的数据用对象表示, 比如, 数据框对象, 时间序列对象, 模型对象, 图形对象, 等等。

S语言表达式可以使用常量和变量。变量名的规则是：由字母、数字、句点组成, 第一个字符必须是字母, 长度没有限制。大小写是不同的。特别要注意句点可以作为名字的合法部分, 而在其它面向对象语言

中句点经常用来分隔对象与成员名。另外, 下划线不能用在名字中, 因为它是赋值符号“<-”的缩写。

6.2.1 常量

常量可以笼统地分为逻辑型、数值型和字符型三种, 实际上数值型数据又可以分为整型、单精度、双精度等, 非特殊需要不必太关心其具体类型。例如, 123, 123.45, 1.2345e30是数值型常量, ”Weight”, ”李明”是字符型(用两个双撇号或两个单撇号包围)。逻辑真值写为T或TRUE(注意区分大小写, 写t或true都没意义), 逻辑假值写为F或FALSE。复数常量就用3.5-2.1i这样的写法表示。

S中的数据可以取缺失值, 用符号NA代表缺失值。函数is.na(x)返回x是否缺失值(真还是假)。

6.2.2 向量(Vector)与赋值

向量是具有相同基本类型的元素序列, 大体相当于其他语言中的一维数组。实际上在S中标量也被看作是长度为1的向量。

定义向量的最常用办法是使用函数c(), 它把若干个数值或字符串组合为一个向量, 比如：

```
> marks <- c(10, 6, 4, 7, 8)
> x <- c(1:3, 10:13)
> x
[1] 1 2 3 10 11 12 13
> x1 <- c(1, 2)
> x2 <- c(3, 4)
> x <- c(x1, x2)
```

```
> x
[1] 1 2 3 4
```

在显示向量值时我们注意到左边总出现一个“[1]”, 这是代表该显示行的第一个数的下标, 例如：

```
> 1234501:1234520
[1] 1234501 1234502 1234503 1234504 1234505 1234506 1234507 1234508 1234509 1234510
[11] 1234511 1234512 1234513 1234514 1234515 1234516 1234517 1234518 1234519 1234520
```

第二行输出从第11个数开始, 所以在行左边显示“[11]”。

S中用符号“<-”(这是小于号紧接一个减号)来为变量赋值。另一种赋值的办法是用assign函数, 比如

```
> x1 <- c(1, 2)
```

和

```
> assign("x1", c(1, 2))
```

效果相同。

函数length(x)可以计算向量x的长度。

6.2.3 向量运算

可以对向量进行加(+)减(−)乘(*)除(/)、乘方(^)运算, 其含意是对向量的每一个元素进行运算。例如：

```
> x <- c(1, 4, 6.25)
> y <- x*2+1
> y
[1] 3.0 9.0 13.5
```

另外, %/%表示整数除法(比如5 %/% 3为1), %%表示求余数(如5 %% 3为2)。

也可以用向量作为函数自变量, sqrt、log、exp、sin、cos、tan等函数都可以用向量作自变量, 结果是对向量的每一个元素取相应的函数值, 如：

```
> sqrt(x)
[1] 1.0 2.0 2.5
```

函数min和max分别取自变量向量的最小值和最大值, 函数sum计算自变量向量的元素和, 函数mean计算均值, 函数var计算样本方差, 函数sd计算标准差(在Splus中用sqrt(var())计算), 函数range返回包含两个值的向量, 第一个值是最小值, 第二个值是最大值。例如：

```
> max(x)
[1] 6.25
```

如果求var(x)而x是$n \times p$矩阵, 则结果为样本协方差阵。

sort(x)返回x的元素从小到大排序的结果向量。order(x)返回使得x从小到大排列的元素下标向量(x[order(x)]等效于sort(x))。

任何数与缺失值的运算结果仍为缺失值。例如,

```
> 2*c(1, NA, 2)
[1] 2 NA 4
> sum(c(1, NA, 2))
[1] NA
```

6.2.4 产生有规律的数列

在S中很容易产生一个等差数列。例如, 1:n产生从1到n的整数列, -2:3产生从-2到3的整数列, 5:2产生反向的数列：

```
> n <- 5
> 1:n
[1] 1 2 3 4 5
> -2:3
[1] -2 -1 0 1 2 3
> 5:2
[1] 5 4 3 2
```

要注意1:n-1不是代表1到n-1而是向量1:n减去1, 这是一个常犯的错误：

```
> 1:n-1
[1] 0 1 2 3 4
> 1:(n-1)
[1] 1 2 3 4
```

seq函数是更一般的等差数列函数。如果只指定一个自变量n>0, 则seq(n)相当于1:n。指定两个自变量时, 第一自变量是开始值, 第二自变量是结束值, 如seq(-2,3)是从-2到3。S函数调用的一个很好的特点是它可以使用不同个数的自变量, 函数可以对不同类型的自变量给出不同结果, 自变量可以用“自变量名=自变量值”的形式指定。例如, seq(-2,3)可以写成seq(from=-2, to=3)。可以用一个by参数指定等差数列的增加值, 例如：

```
> seq(0, 2, 0.7)
[1] 0.0 0.7 1.4
```

也可以写成seq(from=0, to=2, by=0.7)。有参数名的参数的次序任意, 如：

```
> seq(0, by=0.7, to=2)
[1] 0.0 0.7 1.4
```

可以用length参数指定数列长度, 如seq(from=10, length=5)产生10到14。seq函数还可以用一种seq(along=向量名)的格式, 这时只能用这一个参数, 产生该向量的下标序列, 如：

```
> x
[1] 1.00 4.00 6.25
> seq(along=x)
[1] 1 2 3
```

另一个类似的函数是rep, 它可以重复第一个自变量若干次, 例如：

```
> rep(x,3)
[1] 1.00 4.00 6.25 1.00 4.00 6.25 1.00 4.00 6.25
```

第一个参数名为x, 第二个参数名为times(重复次数)。

6.2.5 逻辑向量

向量可以取逻辑值, 如：

```
> l <- c(T, T, F)
> l
[1] TRUE TRUE FALSE
```

当然, 逻辑向量是一个比较的结果, 如：

```
> x
[1] 1.00 4.00 6.25
> l <- x > 3
> l
[1] FALSE TRUE TRUE
```

一个向量与常量比较大小, 结果还是一个向量, 元素为每一对比较的结果逻辑值。

两个向量也可以比较, 如：

```
> log(10*x)
[1] 2.302585 3.688879 4.135167
> log(10*x) > x
[1] TRUE FALSE FALSE
```

比较运算符包括<, <=, >, >=, ==(相等), !=(不等)。

两个逻辑向量可以进行与(&)、或(|)运算, 结果是对应元素运算的结果。对逻辑向量x计算!x表示取每个元素的非。例如：

```
> (x > 1.5) & log(10*x)>1.5
[1] FALSE FALSE TRUE
```

判断一个逻辑向量是否都为真值的函数是all, 如：

```
> all(log(10*x) > x)
[1] FALSE
```

判断是否其中有真值的函数是any, 如：

```
> any(log(10*x) > x)
[1] TRUE
```

函数is.na(x)用来判断x的每一个元素是否缺失。如

```
> is.na(c(1, NA, 3))
[1] FALSE TRUE FALSE
```

逻辑值可以强制转换为整数值, TRUE变成1, FALSE变成0。例如, 我们以age>65为老年人, 否则为年轻人, 可以用c("young", "old")[(age>65)+1]这样的表示。当年龄大于65时age>65等于TRUE, 加1则把TRUE转换为数值型的1, 结果得2, 于是返回第二个下标处的"old"。否则等于0+1下标处的"young"。

6.2.6 字符型向量

向量元素可以取字符串值。例如:

```
> c1 <- c("x", "sin(x)")
> c1
[1] "x" "sin(x)"
> ns <- c("Weight", "Height", "年龄")
> ns
[1] "Weight" "Height" "年龄"
```

paste函数用来把它的自变量连成一个字符串, 中间用空格分开, 例如:

```
> paste("My", "Job")
[1] "My Job"
```

连接的自变量可以是向量, 这时各对应元素连接起来, 长度不相同时较短的向量被重复使用。自变量可以是数值向量, 连接时自动转换成适当的字符串表示, 例如:

```
> paste(c("X", "Y"), "=", 1:4)
[1] "X = 1" "Y = 2" "X = 3" "Y = 4"
```

分隔用的字符可以用sep参数指定, 例如下例产生若干个文件名：

```
> paste("result.", 1:6, sep="")
[1] "result.1" "result.2" "result.3" "result.4" "result.5" "result.6"
```

如果给paste()函数指定了collapse参数, 则paste()可以把一个字符串向量的各个元素连接成一个字符串, 中间用collapse指定的值分隔。比如paste(c('a', 'b'), collapse='.')得到'a.b'。

6.2.7 复数向量

S支持复数运算。复数常量只要用3.5+2.1i这样的格式即可。复向量的每一个元素都是复数。可以用complex()函数生成复向量, 如给定实部和虚部向量定义复数向量：

```
> x <- (0:100)/100*2*pi
> y <- sin(x)
> z <- complex(re=x, im=y)
> plot(z)
```

也可以给定模和辐角定义复数向量：

```
> zz <- complex(mod=rep(1, 10), arg=(0:11)/12*2*pi)
> plot(zz)
```

Re()计算实部, Im()计算虚部, Mod()计算复数模, Arg()计算复数幅角, Conj()计算共轭。基本的数学函数也支持复数运算, 比如：

```
> sqrt(-2+0i)
[1] 0+1.414214i
```

注意自变量是复数时sqrt等函数才进行复数运算。

6.2.8 向量下标运算

S提供了十分灵活的访问向量元素和向量子集的功能。某一个元素只要用x[i]的格式访问, 其中x是一个向量名, 或一个取向量值的表达式, 如：

```
> x
[1] 1.00 4.00 6.25
> x[2]
[1] 4
> (c(1, 3, 5) + 5)[2]
[1] 8
```

可以单独改变一个元素的值, 例如：

```
> x[2] <- 125
> x
[1] 1.00 125.00 6.25
```

事实上, S提供了四种方法来访问向量的一部分, 格式为x[v], x为向量或向量值的表达式, v是如下的表示下标向量：

一、取正整数值的下标向量

在x[v]中, v为一个向量, 元素取值在1到length(x)之间, 取值允许重复, 例如,

```
> x[c(1,3)]
[1] 1.00 6.25
> x[1:2]
[1] 1 125
> x[c(1,3,2,1)]
[1] 1.00 6.25 125.00 1.00
> c("a", "b", "c")[rep(c(2,1,3), 3)]
[1] "b" "a" "c" "b" "a" "c" "b" "a" "c"
```

二、取负整数值的下标向量

在x[v]中, v为一个向量, 元素取值在−length(x)到−1之间, 表示扣除相应位置的元素。例如：

```
> x[-(1:2)]
[1] 6.25
```

三、取逻辑值的下标向量

在x[v]中, v为和x等长的逻辑向量, x[v]表示取出所有v为真值的元素, 如：

```
> x
[1] 1.00 125.00 6.25
> x<10
[1] TRUE FALSE TRUE
> x[x<10]
[1] 1.00 6.25
> x[x<0]
numeric(0)
```

可见x[x<10]取出所有小于10的元素组成的子集。这种逻辑值下标是一种强有力的检索工具, 例如x[sin(x)>0]可以取出x中所有正弦函数值为正的元素组成的向量。

如果下标都是假值则结果是一个零长度的向量, 显示为numeric(0)。

四、取字符型值的下标向量

在定义向量时可以给元素加上名字, 例如：

```
> ages <- c(Li=33, Zhang=29, Liu=18)
> ages
   Li Zhang   Liu
   33    29    18
```

这样定义的向量可以用通常的办法访问, 另外还可以用元素名字来访问元素或元素子集, 例如：

```
> ages["Zhang"]
Zhang
   29
> ages[c("Li", "Liu")]
 Li Liu
 33  18
```

向量元素名可以后加, 例如：

```
> ages1 <- c(33, 29, 18)
> names(ages1) <- c("Li", "Zhang", "Liu")
> ages1
   Li Zhang   Liu
   33    29    18
```

上面我们看到了如何访问向量的部分元素。在S中还可以改变一部分元素的值, 例如：

```
> x
[1] 1.00 125.00 6.25
> x[c(1,3)] <- c(144, 169)
> x
[1] 144 125 169
```

注意赋值的长度必须相同, 例外是可以把部分元素赋为一个统一值：

```
> x[c(1,3)] <- 0
> x
[1] 0 125 0
```

要把向量所有元素赋为一个相同的值而又不想改变其长度, 可以用`x[]`的写法：

```
> x[] <- 0
```

注意这与“`x <- 0`”是不同的, 前者赋值后向量长度不变, 后者使向量变为标量0。

改变部分元素值的技术与逻辑值下标方法结合可以定义向量的分段函数, 例如, 要定义y=f(x)为当x<0时取1−x, 否则取1+x, 可以用：

```
> y <- numeric(length(x))
> y[x<0] <- 1 - x[x<0]
> y[x>=0] <- 1 + x[x>=0]
```

6.3 多维数组和矩阵

6.3.1 数组(array)和矩阵(matrix)

数组(array)可以看成是带多个下标的类型相同的元素的集合, 常用的是数值型的数组如矩阵, 也可以有其它类型(如字符型、逻辑型、复型数组)。S可以很容易地生成和处理数组, 特别是矩阵(二维数组)。

数组有一个特征属性叫做维数向量(dim属性), 维数向量是一个元素取正整数值的向量, 其长度是数组的维数, 比如维数向量有两个元素时数组为二维数组(矩阵)。维数向量的每一个元素指定了该下标的上界, 下标的下界总为1。

一组值只有定义了维数向量(dim属性)后才能被看作是数组。比如：

```
> z <- 1:1500
> dim(z) <- c(3, 5, 100)
```

这时z已经成为了一个维数向量为c(3,5,100)的三维数组。也可以把向量定义为一维数组, 例如：

```
> dim(z) <- 1500
```

数组元素的排列次序缺省情况下是采用FORTRAN的数组元素次序(按列次序), 即第一下标变化最快, 最后下标变化最慢, 对于矩阵(二维数组)则是按列存放。例如, 假设数组a的元素为1:24, 维数向量为c(2,3,4), 则各元素次序为a[1,1,1], a[2,1,1], a[1,2,1], a[2,2,1], a[1,3,1], ..., a[2,3,4]。

用函数array()或matrix()可以更直观地定义数组。array()函数的完全使用为array(x, dim=length(x), dimnames=NULL), 其中x是第一自变量, 应该是一个向量, 表示数组的元素值组成的向量。dim参数可省, 省略时作为一维数组(但不同于向量)。dimnames属性可以省略, 不省略时是一个长度与维数相同的列表(list, 见后面), 列表的每个成员为一维的名字。例如上面的z可以这样定义：

```
> z <- array(1:1500, dim=c(3,5,100))
```

函数matrix()用来定义最常用的一种数组：二维数组, 即矩阵。其完全格式为

```
matrix(data = NA, nrow = 1, ncol = 1,
       byrow = FALSE, dimnames = NULL)
```

其中第一自变量data为数组的数据向量(缺省值为缺失值NA), nrow为行数, ncol为列数, byrow表示数据填入矩阵时按行次序还是列次序, 一定注意缺省情况下按列次序, 这与我们写矩阵的习惯是不同的。dimnames缺省是空值, 否则是一个长度为2的列表, 列表第一个成员是长度与行数相等的字符型向量, 表示每行的标签, 列表第二个成员是长度与列数相同的字符型向量, 表示每列的标签。例如, 定义一个3行4列, 由1:12按行次序排列的矩阵, 可以用：

```
> b <- matrix(1:12, ncol=4, byrow=T)
> b
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
```

```
[2,]    5    6    7    8
[3,]    9   10   11   12
```

注意在有数据的情况下只需指定行数或列数之一。指定的数据个数允许少于所需的数据个数, 这时循环使用提供的数据。例如：

```
> b <- matrix(0, nrow=3, ncol=4)
```

生成3行4列的元素都为0的矩阵。

6.3.2 数组下标

要访问数组的某个元素, 只要象上面那样写出数组名和方括号内用逗号分开的下标即可, 如a[2,1,2]。

更进一步我们还可以在每一个下标位置写一个下标向量, 表示对这一维取出所有指定下标的元素, 如a[1, 2:3, 2:3]取出所有第一下标为1, 第二下标为2或3, 第三下标为2或3的元素。注意因为第一维只有一个下标所以退化了, 得到的是一个维数向量为c(2,2)的数组。

另外, 如果略写某一维的下标, 则表示该维全选。例如, `a[1, , ]`取出所有第一下标为1的元素, 得到一个形状为c(3,4)的数组。`a[ , 2, ]`取出所有第二下标为2的元素得到一个形状为c(2,4)的数组。`a[1,1, ]`则只能得到一个长度为4的向量, 不再是数组(`dim(a[1,1, ])`值为NULL)。`a[ , , ]`或`a[]`都表示整个数组。比如

```
a[] <- 0
```

可以在不改变数组维数的条件下把元素都赋成0。

还有一种特殊下标是对于数组只用一个下标向量(是向量, 不是数组), 比如a[3:4], 这时忽略数组的维数信息, 把下标表达式看作是对数组的数据向量取子集。

6.3.3 不规则数组下标

在S中甚至可以把数组中的任意位置的元素作为一组访问, 其方法是用一个矩阵作为数组的下标, 矩阵的每一行是一个元素的下标, 数组有几维下标矩阵的每一行就有几列。例如, 我们要把上面的形状为c(2,3,4)的数组a的第[1,1,1], [2,2,3], [1,3,4], [2,1,4]号共四个元素作为一个整体访问, 先定义一个包含这些下标作为行的二维数组:

```
> b <- matrix(c(1,1,1, 2,2,3, 1,3,4, 2,1,4),
       ncol=3, byrow=T)
> b
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    2    2    3
[3,]    1    3    4
[4,]    2    1    4
> a[b]
[1] 1 16 23 20
```

注意取出的是一个向量。我们还可以对这几个元素赋值, 如:

```
> a[b] <- c(101, 102, 103, 104)
> a
```

或

```
> a[b] <- 0
> a
```

6.3.4 数组四则运算

数组可以进行四则运算(+, −, *, /, ^), 解释为数组对应元素的四则运算, 参加运算的数组一般应该是相同形状的(dim属性完全相同)。例如, 假设A, B, C是三个形状相同的数组, 则

```
> D <- C + 2*A/B
```

计算得到的结果是A的每一个元素除以B的对应元素再乘以2然后加上C的对应元素得到相同形状的数组。四则运算遵循通常的优先级规则。

形状不一致的向量和数组在少数情况下也可以进行四则运算, 一般的规则是数组的数据向量对应元素进行运算, 把短的循环使用来与长的匹配, 并尽可能保留共同的数组属性。例如：

```
> x1 <- c(100, 200)
> x2 <- 1:6
> x1+x2
[1] 101 202 103 204 105 206
> x3 <- matrix(1:6, nrow=3)
> x3
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> x1+x3
     [,1] [,2]
```

```
[1,]  101  204
[2,]  202  105
[3,]  103  206
```

除非你清楚地知道规则应避免使用这样的办法(标量与数组或向量的四则运算除外)。

6.3.5 矩阵运算

矩阵是二维数组, 但因为其应用广泛所以对它定义了一些特殊的运算和操作。

函数t(A)返回矩阵A的转置。nrow(A)为矩阵A的行数, ncol(A)为矩阵A的列数。

矩阵之间进行普通的加减乘除四则运算仍遵从一般的数组四则运算规则, 即数组的对应元素之间进行运算, 所以注意A*B不是矩阵乘法而是矩阵对应元素相乘。

要进行矩阵乘法, 使用运算符**%*%**, `A %*% B`表示矩阵A乘以矩阵B(当然要求A的列数等于B的行数)。例如：

```
> A <- matrix(1:12, nrow=4, ncol=3, byrow=T)
> B <- matrix(c(1,0), nrow=3, ncol=2, byrow=T)
> A
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
> B
     [,1] [,2]
[1,]    1    0
[2,]    1    0
[3,]    1    0
```

```
> A %*% B
     [,1] [,2]
[1,]    6    0
[2,]   15    0
[3,]   24    0
[4,]   33    0
>
```

另外, 向量用在矩阵乘法中可以作为行向量看待也可以作为列向量看待, 这要看哪一种观点能够进行矩阵乘法运算。例如, 设x是一个长度为n的向量, A是一个$n \times n$矩阵, 则"**x %*% A %*% x**"表示二次型$x'Ax$。但是, 有时向量在矩阵乘法中的地位并不清楚, 比如"**x %*% x**"就既可能表示内积$x'x$也可能表示$n \times n$阵$x'x$。因为前者较常用, 所以S选择表示前者, 但内积最好还是用crossprod(x)来计算。要表示xx', 可以用"cbind(x) **%*%** x"或"x **%*%** rbind(x)"。

函数crossprod(X, Y)表示一般的交叉乘积(内积)$X'Y$, 即X的每一列与Y的每一列的内积组成的矩阵。如果X和Y都是向量则是一般的内积。只写一个参数X的crossprod(X)计算X自身的内积$X'X$。

其它矩阵运算还有solve(A,b)解线性方程组$Ax = b$, solve(A)求方阵A的逆矩阵, svd()计算奇异值分解, qr()计算QR分解, eigen()计算特征向量和特征值。详见随机帮助, 例如**:**

```
> ?qr
```

函数diag()的作用依赖于其自变量。diag(vector)返回以自变量(向量)为主对角元素的对角矩阵。diag(matrix)返回由矩阵的主对角元素组成的向量。diag(k)(k为标量)返回k阶单位阵。

6.3.6 矩阵合并与拉直

函数cbind()把其自变量横向拼成一个大矩阵, rbind()把其自变量纵向拼成一个大矩阵。cbind()的自变量是矩阵或者看作列向量的向量, 自变量的高度应该相等(对于向量, 高度即长度, 对于矩阵, 高度即行数)。rbind的自变量是矩阵或看作行向量的向量, 自变量的宽度应该相等(对于向量, 宽度即长度, 对于矩阵, 宽度即列数)。如果参与合并的自变量比其它自变量短则循环补足后合并。例如:

```
> x1 <- rbind(c(1,2), c(3,4))
> x1
     [,1] [,2]
[1,]    1    2
[2,]    3    4
> x2 <- 10+x1

> x3 <- cbind(x1, x2)

> x3
     [,1] [,2] [,3] [,4]
[1,]    1    2   11   12
[2,]    3    4   13   14
> x4 <- rbind(x1, x2)
> x4
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]   11   12
[4,]   13   14
> cbind(1, x1)
     [,1] [,2] [,3]
[1,]    1    1    2
[2,]    1    3    4
```

因为cbind()和rbind()的结果总是矩阵类型(有dim属性且为二维), 所以可以用它们把向量表示为$n \times 1$矩阵(用cbind(x))或$1 \times n$矩阵(用rbind(x))。

设a是一个数组, 要把它转化为向量(去掉dim和dimnames属性), 只要用函数as.vector(a)返回值就可以了(注意函数只能通过函数值返回结果而不允许修改它的自变量, 比如t(X)返回X的转置矩阵而X本身并未改变)。另一种由数组得到其数据向量的简单办法是使用函数c(), 例如c(a)的结果是a的数据向量。这样的另一个好处是c()允许多个自变量, 它可以把多个自变量都看成数据向量连接起来。例如, 设A和B是两个矩阵, 则c(A,B)表示把A按列次序拉直为向量并与把B按列次序拉直为向量的结果连接起来。一定注意拉直时是按列次序拉直的。

6.3.7 数组的维名字

数组可以有一个属性dimnames保存各维的各个下标的名字, 缺省时为NULL(即无此属性)。我们以矩阵为例, 它有两个维：行维和列维。比如, 设x为2行3列矩阵, 它的行维可以定义一个长度为2的字符向量作为每行的名字, 它的列维可以定义一个长度为3的向量作为每列的名字, 属性dimnames是一个列表, 列表的每个成员是一个维名字的字符向量或NULL。例如：

```
> x <- matrix(1:6, ncol=2,
+ dimnames=list(c("one", "two", "three"),
+                c("First", "Second")),
+ byrow=T)
> x
```

```
      First Second
one       1      2
two       3      4
three     5      6
```

我们也可以先定义矩阵x然后再为dimnames(x)赋值。

对于矩阵, 我们还可以使用属性rownames和colnames来访问行名和列名。如：

```
> x <- matrix(1:6, ncol=2,byrow=T)
> colnames(x) <- c("First", "Second")
> rownames(x) <- c("one", "two", "three")
```

在定义了数组的维名后我们对这一维的下标就可以用它的名字来访问, 例如：

```
> x[c("one","three"),]
      First Second
one       1      2
three     5      6
```

6.3.8 数组的外积

两个数组a和b的外积是由a的每一个元素与b的每一个元素搭配在一起相乘得到一个新元素, 这样得到一个维数向量等于a的维数向量与b的维数向量连起来的数组, 即若d为a和b的外积, 则dim(d)=c(dim(a), dim(b))。

a和b的外积记作`a %o% b`。如

```
> d <- a %o% b
```

也可以写成一个函数调用的形式：

```
> d <- outer(a, b, '*')
```

注意outer(a, b, f)是一个一般性的外积函数, 它可以把a的任一个元素与b的任意一个元素搭配起来作为f的自变量计算得到新的元素值, 外积是两个元素相乘的情况。函数当然也可以是加、减、除, 或其它一般函数。当函数为乘积时可以省略不写。

例如, 我们希望计算函数$z = \cos(y)/(1 + x^2)$在一个x和y的网格上的值用来绘制三维曲面图, 可以用如下方法生成网格及函数值：

```
> x <- seq(-2, 2, length=20)
> y <- seq(-pi, pi, length=20)
> f <- function(x, y) cos(y)/(1+x^2)
> z <- outer(x, y, f)
> persp(x,y,z)
```

用这个一般函数可以很容易地把两个数组的所有元素都两两组合一遍进行指定的运算。

下面考虑一个有意思的例子。我们考虑简单的2×2矩阵$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$, 其元素均在0,1,...,9中取值。假设四个元素a, b, c, d都是相互独立的服从离散均匀分布的随机变量, 我们设法求矩阵行列式$ad - bc$的分布。首先, 随机变量ad和bc同分布, 它的取值由以下外积矩阵给出, 每一个取值的概率均为1/100:

```
> d <- outer(0:9, 0:9)
```

这个语句产生一个10×10的外积矩阵。

为了计算ad的100个值(有重复)与bc的100个值相减得到的10000个结果, 可以使用如下外积函数：

```
> d2 <- outer(d, d, "-")
```

这样得到一个维数向量为c(10,10,10,10)的四维数组, 每一个元素为行列式的一个可能取值, 概率为万分之一。因为这些取值中有很多重复, 我们可以用一个table()函数来计算每一个值的出现次数(频数)：

```
      > fr <- table(d2)
```

得到的结果是一个带有元素名的向量fr, fr的元素名为d2的一个取值, fr的元素值为d2该取值出现的频数。比如fr[1]的元素名为-81, 值为19, 表示值-81在数组d2中出现了19次。通过计算length(fr)可以知道共有163个不同值。还可以把这些值绘制一个频数分布图(除以10000则为实际概率), 见图6.2：

```
> plot(as.numeric(names(fr)), fr, type="h",
+ xlab="Determinant", ylab="Frequency")
```

其中as.numeric()把向量fr中的元素名又转换成了数值型, 用来作为作图的横轴坐标, fr中的元素值即频数作为纵轴, type=”h”表示是画垂线型图。

6.3.9 数组的广义转置

可以用aperm(a, perm)函数把数组a的各维按perm中指定的新次序重新排列。例如：

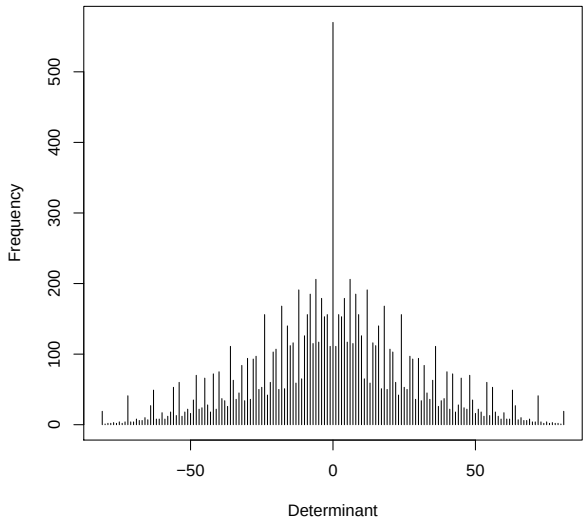


图 6.2: 随机矩阵行列式分布

```
> a <- array(1:24, dim=c(2,3,4))
> b <- aperm(a, c(2, 3, 1))
```

结果a的第2维变成了b的第1维, a的第3维变成了b的第2维, a的第1维变成了b的第3维。这时有a[i1, i2, i3]≡b[i2, i3, i1]。注意c(i1, i2, i3)[2,3,1]=c(i2, i3, i1)。一般地, 若b <- aperm(a, p), i是数组a的一个下标向量, 则a[rbind(i)]≡b[rbind(i[p])], 即a的一个元素下标经过p变换后成为b的对应元素的下标。

对于矩阵a, aperm(a, c(2,1))恰好是矩阵转置。对于矩阵转置可以简单地用t(a)表示。

6.3.10 apply函数

对于向量, 我们有sum、mean等函数对其进行计算。

对于数组, 如果我们想对其中一维(或若干维)进行某种计算, 可以用apply函数。其一般形式为：

```
apply(X, MARGIN, FUN, ...)
```

其中X为一个数组, MARGIN是固定哪些维不变, FUN是用来计算的函数。例如, 设a是3 × 4矩阵, 则apply(a, 1, sum)的意义是对a的各行求和(保留第一维即第一个下标不变), 结果是一个长度为3的向量(与第一维长度相同), 而apply(a, 2, sum)意义是对a的各列求和, 结果是一个长度为4的向量(与第二维长度相同)。

如果函数FUN的结果是一个标量, MARGIN只有一个元素, 则apply的结果是一个向量, 其长度等于MARGIN指定维的长度, 相当于固定MARGIN指定的那一维的每一个值而把其它维取出作为子数组或向量送入FUN中进行运算。如果MARGIN指定了多个维, 则结果是一个维数向量等于dim(X)[MARGIN]的数组。如果函数FUN的结果是一个长度为N的向量, 则结果是一个维数向量等于c(N, dim(X)[MARGIN])的数组, 注意这时不论是对哪一维计算, 结果都放在了第一维。所以, 比如我们要把4×3矩阵a的3列分别排序, 只要用apply(a, 2, sort), 这样对每一列排序得到一个长度为4的向量, 用第一维来引用, 结果的维向量为c(N, dim(a)[2])=c(4,3), 保留了列维, 恰好得到所需结果, 运行如下例：

```
> a <- cbind(c(4,9,1), c(3,7,2))
> a
```

```
     [,1] [,2]
[1,]    4    3
[2,]    9    7
[3,]    1    2
> apply(a, 2, sort)
     [,1] [,2]
[1,]    1    2
[2,]    4    3
[3,]    9    7
>
```

但是, 如果要对行排序, 则apply(a, 1, sort)把a的每一行3个元素排序后的结果用第一维来引用, 结果的维向量为c(N, dim(a)[1])=c(3, 4), 把原来的列变成了行, 所以t(apply(a,1,sort))才是对a的每一行排序的结果。如：

```
> apply(a, 1, sort)
     [,1] [,2] [,3]
[1,]    3    7    1
[2,]    4    9    2
> t(apply(a,1,sort))
     [,1] [,2]
[1,]    3    4
[2,]    7    9
[3,]    1    2
```

上面我们只用了矩阵(二维数组)作为例子讲解apply的用法。实际上, apply可以用于任意维数的数组, 函数FUN也可以是任意可以接收一个向量或数组作为第一自变量的函数。比如, 设x是一个维数向量为c(2,3,4,5)的数组, 则apply(x, c(1,3), sum)可以产生一个2行4列的矩阵, 其每一元素是x中固定第1维和第3维下标取出子数组求和的结果。

6.4 因子

统计中的变量有两个重要类别：区间变量和名义变量、有序变量。区间变量取连续的数值, 可以进行求和、平均等运算。名义变量和有序变量取离散值, 可以用数值代表也可以是字符型值, 其具体数值没有加减乘除的意义, 不能用来计算而只能用来分类或者计数。名义变量比如性别、省份、职业, 有序变量比如班级名次。

因为离散变量有各种不同表示方法, 在S中为统一起见使用因子(factor)来表示这种分类变量。还提供了有序因子(ordered factor)来表示有序变量。因子是一种特殊的向量, 其中每一个元素取一组离散值中的一个, 因子对象有一个特殊属性levels表示这组离散值。例如：

```
> x <- c("男", "女", "男", "男",  "女")
> y <- factor(x)
> y
[1] 男 女 男 男 女
Levels:  男 女
```

函数factor()用来把一个向量编码成为一个因子。其一般形式为：

```
factor(x, levels = sort(unique(x), na.last = TRUE),
       labels, exclude = NA, ordered = FALSE)
```

可以自行指定各离散取值(水平levels), 不指定时由x的不同值来求得。labels可以用来指定各水平的标签, 不指定时用各离散取值的对应字符

串。exclude参数用来指定要转换为缺失值(NA)的元素值集合。如果指定了levels, 则因子的第i个元素当它等于水平中第j个时元素值取"j", 如果它的值没有出现在levels中则对应因子元素值取NA。ordered取真值时表示因子水平是有次序的(按编码次序)。例如：

```
> f <- factor(c(1,0,1,1,0),
       levels=c(1,0), labels=c("男","女"))
```

结果和刚才一样。

可以用is.factor()检验对象是否因子, 用as.factor()把一个向量转换成一个因子。

因子的基本统计是频数统计, 用函数table()来计数。例如,

```
> sex <- factor(c("男", "女", "男", "男", "女"))
> res.tab <- table(sex)
> res.tab
sex
男 女
 3  2
```

表示男性3人, 女性2人。table()的结果是一个带元素名的向量, 元素名为因子水平, 元素值为该水平的出现频数。

从这个简单例子我们也可以看出S与SAS的一个区别：SAS的结果一般只显示出来, 保存只能存为数据集一种类型, 不便于对中间结果进一步编程处理；而S的结果除了可以显示外本身都是S对象(如这里的向量结果), 可以很方便地进一步处理。

可以用两个或多个因子进行交叉分类。比如, 性别(sex)和职业(job)交叉分组可以用table(sex, job)来统计每一交叉类的频数, 结果为一个矩阵, 矩阵带有行名和列名, 分别为两个因子的各水平名。

因子可以用来作为另外的同长度变量的分类变量。比如, 假设上面的sex是5个学生的性别, 而

```
> h <- c(165, 170, 168, 172, 159)
```

是这5个学生的身高, 则

```
> tapply(h, sex, mean)
```

可以求按性别分类的身高平均值。这样用一个等长的因子向量对一个数值向量分组的办法叫做不规则数组(ragged array)。后面我们还可以看到更多的因子的应用。

6.5 列表(list)

6.5.1 列表定义

列表是一种特别的对象集合, 它的元素也由序号(下标)区分, 但是各元素的类型可以是任意对象, 不同元素不必是同一类型。元素本身允许是其它复杂数据类型, 比如, 列表的一个元素也允许是列表。这样的数据类型称为是递归(recursive)数据类型。例如:

```
> rec <- list(name="李明", age=30, scores=c(85, 76, 90))
> rec
$name
[1] "李明"
```

```
$age
[1] 30

$scores
[1] 85 76 90
```

列表元素总可以用“列表名[[下标]]”的格式引用。例如：

```
> rec[[2]]
[1] 30
> rec[[3]][2]
[1] 76
```

但是, 列表不同于向量, 我们每次只能引用一个元素, 如rec[[1:2]]的用法是不允许的。

注意：“列表名[下标]”或“列表名[下标范围]”的用法也是合法的, 但其意义与用两重括号的记法完全不同, 两重记号取出列表的一个元素, 结果与该元素类型相同, 如果使用一重括号, 则结果是列表的一个子列表(结果类型仍为列表)。

在定义列表时如果指定了元素的名字(如rec中的name, age, scores), 则引用列表元素还可以用它的名字作为下标, 格式为“列表名[[”元素名”]]”, 如：

```
> rec[["age"]]
[1] 30
```

另一种格式是“列表名$元素名”, 如：

```
> rec$age
[1] 30
```

其中“元素名”可以简写到与其它元素名能够区分的最短程度, 比如“rec$s”可以代表“rec$score”。这种写法方便了交互运行, 编写程序时一般不用简写以免降低程序的可读性。

使用元素名的引用方法可以让我们不必记住某一个下标代表那一个元素, 而直接用易记的元素名来引用元素。事实上, 向量和矩阵也可以指定元素名、行名、列名。

定义列表使用list()函数, 每一个自变量变成列表的一个元素, 自变量可以用“名字=值”的方式给出, 即给出列表元素名。自变量的值被复制到列表元素中, 自变量如果是变量并不会与该列表元素建立关系(改变该列表元素不会改变自变量的值)。

6.5.2 修改列表

列表的元素可以修改, 只要把元素引用赋值即可。如：

```
> rec$age <- 45
```

甚至

```
> rec$age <- list(19, 29, 31)
```

(可以任意修改一个列表元素)。如果被赋值的元素原来不存在, 则列表延伸以包含该新元素。例如, rec现在共有三个元素, 我们定义一个新的命名元素, 则列表长度变为4, 再定义第六号元素则列表长度变为6：

```
> rec$sex <- "男"
> rec[[6]] <- 161
> rec
$name
[1] "李明"

$age
[1] 30

$scores
[1] 85 76 90

$sex
[1] "男"

[[5]]
NULL

[[6]]
[1] 161
```

第五号元素因为没有定义所以其值是“NULL”, 这是空对象的记号。如果rec是一个向量, 则其空元素为“NA”, 这是缺失值的记号。从这里我们也可以体会“NULL”与“NA”的区别。

几个列表可以用连接函数c()连接起来, 结果仍为一个列表, 其元素为各自变量的列表元素。如：

```
> list.ABC <- c(list.A, list.B, list.C)
```

注意在S中句点是名字的合法部分, 一般没有特殊意义。

6.5.3 几个返回列表的函数

列表的重要作用是把相关的若干数据保存在一个数据对象中, 这样在编写函数时我们就可以返回这样一个包含多项输出的列表。因为函数的返回结果可以完整地存放在一个列表中, 我们可以继续对得到的结果进行分析, 这是S比SAS灵活的一个地方。下面给出几个返回列表的例子。

一、特征值和特征向量

函数eigen(x)对对称矩阵x计算其特征值和特征向量, 返回结果为一个列表, 列表的两个成员(元素)为values和vectors。例如:

```
> ev <- eigen((1:3) %o% (1:3))
> ev
$values
[1]  1.400000e+01  0.000000e+00 -8.881784e-16

$vectors
          [,1]       [,2]       [,3]
[1,] 0.2672612  0.8944272 -0.3585686
[2,] 0.5345225 -0.4472136 -0.7171372
[3,] 0.8017837  0.0000000  0.5976143
```

可见三个特征值只有第一个不为零(由于数值计算精度所限, 第三个特征值应为零但结果只是近似为零)。特征向量按矩阵存放, 每一列为一个特征向量。

二、奇异值分解及行列式

函数svd()进行奇异值分解$X = UDV'$, 其中X是任

意$n \times m$阵, U为$n \times n$正交阵, V为$m \times m$正交阵, D为$n \times m$对角阵(只有主对角线元素不为零且非负)。svd(x)返回有三个成员d, u, v的列表, d为包含奇异值的向量(即D的主对角线元素), u, v分别为上面的两个正交阵。易见如果矩阵x是方阵则x的行列式的绝对值等于奇异值的乘积, 所以：

```
> absdetx <- prod(svd(x)$d)
```

或者我们可以为此定义一个函数：

```
> absdet <- function(x) prod(svd(x)$d)
```

三、最小二乘拟合与QR分解

函数lsfit(x,y)返回最小二乘拟合的结果。最小二乘的模型为线性模型

$$y = X\beta + \varepsilon$$

lsfit(x,y)的第一个参数x为模型中的设计阵X, 第二个参数y为模型中的因变量y(可以是一个向量也可以是一个矩阵), 返回一个列表, 成员coefficients为上面模型的β(最小二乘系数), 成员residuals为拟合残差, 成员intercept用来指示是否有截距项, 成员qr为设计阵X的QR分解, 它本身也是一个列表。模型拟合缺省情况有截距项, 可以用intercept=FALSE选项指定无截距项。关于最小二乘拟合还可参见ls.diag()函数(查看帮助)。

函数qr(x)返回x的QR分解结果。矩阵X的QR分解为$X = QR$, Q为正交阵, R为上三角阵。函数结果为

一个列表, 成员qr是压缩了Q和R在内的一个矩阵, 包括对角线在内的上三角部分为R, Q的信息压缩存放在下三角部分。结果的其它成员为一些辅助信息。用qr.Q()和qr.R()可以从qr()的结果中提取Q和R。

6.6 数据框(data.frame)

数据框是S中类似SAS数据集的一种数据结构。它通常是矩阵形式的数据, 但矩阵各列可以是不同类型的。数据框每列是一个变量, 每行是一个观测。

但是, 数据框有更一般的定义。它是一种特殊的列表对象, 有一个值为"data.frame"的class属性, 各列表成员必须是向量(数值型、字符型、逻辑型)、因子、数值型矩阵、列表, 或其它数据框。向量、因子成员为数据框提供一个变量, 如果向量非数值型则会被强制转换为因子, 而矩阵、列表、数据框这样的成员为新数据框提供了和其列数、成员数、变量数相同个数的变量。作为数据框变量的向量、因子或矩阵必须具有相同的长度(行数)。

尽管如此, 我们一般还是可以把数据框看作是一种推广了的矩阵, 它可以用矩阵形式显示, 可以用对矩阵的下标引用方法来引用其元素或子集。

6.6.1 数据框生成

数据框可以用data.frame()函数生成, 其用法与list()函数相同, 各自变量变成数据框的成分, 自变量可以命名, 成为变量名。例如:

```
> d <- data.frame(name=c("李明", "张聪", "王建"),
+ age=c(30, 35, 28), height=c(180, 162, 175))
> d
  name age height
1 李明  30    180
2 张聪  35    162
3 王建  28    175
```

如果一个列表的各个成分满足数据框成分的要求, 它可以用as.data.frame()函数强制转换为数据框。比如, 上面的d如果先用list()函数定义成了一个列表, 就可以强制为一个数据框。

一个矩阵可以用data.frame()转换为一个数据框, 如果它原来有列名则其列名被作为数据框的变量名, 否则系统自动为矩阵的各列起一个变量名(如X1, X2)。

6.6.2 数据框引用

引用数据框元素的方法与引用矩阵元素的方法相同, 可以使用下标或下标向量, 也可以使用名字或名字向量。如d[1:2, 2:3]。数据框的各变量也可以用按列表引用(即用双括号[[]]或$符号引用)。

数据框的变量名由属性names定义, 此属性一定是非空的。数据框的各行也可以定义名字, 可以用rownames属性定义。如:

```
> names(d)
[1] "name" "age" "height"
> rownames(d)
[1] "1" "2" "3"
```

6.6.3 attach()函数

数据框的主要用途是保存统计建模需要的数据。S的统计建模功能都需要以数据框为输入数据。我们也可以把数据框当成一种矩阵来处理。

在使用数据框的变量时可以用"数据框名$变量名"的记法。但是, 这样使用较麻烦, S提供了attach()函数可以把数据框"连接"入当前的名字空间。例如,

```
> attach(d)
> r <- height / age
```

后一语句将在当前工作空间建立一个新变量r, 它不会自动进入数据框d, 要把新变量赋值到数据框中, 可以用

```
> d$r <- height / age
```

这样的格式。

为了取消连接, 只要调用detach()(无参数即可)。

注意：S和R中名字空间的管理是比较独特的。它在运行时保持一个变量搜索路径表, 在读取某个变量时到这个变量搜索路径表中由前向后查找, 找到最前的一个；在赋值时总是在位置1赋值(除非特别指定在其它位置赋值)。attach()的缺省位置是在变量搜索路径表的位置2, detach()缺省也是去掉位置2。所以, S编程的一个常见问题是当你误用了一个自己并没有赋值的变量时有可能不出错, 因为这个变量已在搜索路径中某个位置有定义, 这样不利于程序的调试, 需要留心这样的问题。

除了可以连接数据框, 也可以连接列表。

6.7 输入输出

6.7.1 输出

在S交互运行时要显示某一个对象的值只要键入其名字即可, 如：

```
> x <- 1:10
> x
[1] 1 2 3 4 5 6 7 8 9 10
```

这实际上是调用了print()函数, 即print(x)。在非交互运行(程序)中应使用print()来输出。print()函数可以带一个digits=参数指定每个数输出的有效数字位数, 可以带一个quote=参数指定字符串输出时是否带两边的撇号, 可以带一个print.gap=参数指定矩阵或数组输出时列之间的间距。

print()函数是一个通用函数, 即它对不同的自变量有不同的反应。对各种特殊对象如数组、模型结果等都可以规定print的输出格式。

cat()函数也用来输出, 但它可以把多个参数连接起来再输出(具有paste()的功能)。例如：

```
> cat("i = ", i, "\n")
```

注意使用cat()时要自己加上换行符"\n"。它把各项转换成字符串, 中间隔以空格连接起来, 然后显示。如果要使用自定义的分隔符, 可以用sep=参数, 例如：

```
> cat(c("AB", "C"), c("E", "F"), "\n", sep="")
ABCDEF
```

cat()还可以指定一个参数“file=文件名”, 可以把结果写到指定的文件中, 如：

```
> cat("i = ", 1, "\n", file="c:/work/result.txt")
```

如果指定的文件已经存在则原来内容被覆盖。加上一个append=TRUE参数可以不覆盖原文件而是在文件末尾附加, 这很适用于运行中的结果记录。

cat()函数和print()都不具有很强的自定义格式功能, 为此可以使用cat()与format()函数配合实现。format()函数为一个数值向量找到一种共同的显示格式然后把向量转换为字符型。例如：

```
> format(c(1, 100, 10000))
[1] "    1" "  100" "10000"
```

S-PLUS中的format()函数功能较强, 具有较多的控制参数, 请参见帮助。R中目前format()函数功能仍较弱, 但R有一个formatC函数可以提供类似C语言的printf格式功能。formatC对输入向量的每一个元素单独进行格式转换而不生成统一格式, 例如：

```
> formatC(c(1, 10000))
[1] "1"      "1e+004"
```

在formatC()函数中可以用format=参数指定C格式类型, 如”d”(整数), ”f”(定点实数), ”e”, ”E”(科学记

数法), "g","G"(选择位数较少的输出格式), "fg"(定点实数但用参数digits指定有效位数而不是总宽度), "s"(字符串)。可以用width指定输出宽度, 用digits指定有效位数(格式为e,E,g,G,fg时)或小数点后位数(格式为f)时。可以用flag参数指定一个输出选项字符串, 字符串中有"-"表示输出左对齐, 有"0"表示左空白用0填充, 有"+"表示要输出正负号, 等等。例如, 我们有一个矩阵da中保存了三个日期的年、月、日：

```
> da
     [,1] [,2] [,3]
[1,]   99    1    3
[2,]   96   11    9
[3,]   65    5   18
```

为了输出这三个日期, 可以用apply函数指定对每一行作用一个输出函数, 此输出函数利用cat()和formatC来控制：

```
> apply(da, 1, function(r)
+ cat(formatC(r[1], format='d', width=2, flag='0'), '-',
+ formatC(r[2], format='d', width=2, flag='0'), '-',
+ formatC(r[3], format='d', width=2, flag='0'), '\n', sep=''))
99-01-03
96-11-09
65-05-18
NULL
```

这里我们知道apply函数第一个参数指定了一个矩阵, 第二个参数说明对行操作还是对列操作, 第三个参数是一个函数, 这里我们使用了直接定义一个函数作为参数的办法。输出结果中多了一个NULL函数, 这是因为我们在交互运行, apply的结果作为一个表达式的值(NULL)会被显示出来。为避免显示, 可以

把结果赋给一个临时变量名, 或者把整个表达式作为invisible()函数的参数, 这时不显示表达式值。

S的输出缺省显示在交互窗口。可以用sink()函数指定一个文件以把后续的输出转向到这个文件, 并可用append参数指定是否要在文件末尾附加：

```
> sink("c:/work/result.txt", append=TRUE)
> ls()
> d
> sink()
```

调用无参数的sink()把输出恢复到交互窗口。

如果要把一个矩阵x输出到文件中, 可以用write(t(x), file="文件名", ncol=ncol(x))的形式。这里之所以要把x转置后再输出是因为R中矩阵是列优先的, 如果不转置则输出是按列输出的。如果不指定列数则缺省使用5列, 即使x的列数多于或少于5。文件名缺省使用"data"。

要把一个数据框d输出到文件中, 可以用write.table(x, file="文件名"), 输出文件中包含变量名表头和行名。

6.7.2 输入

为了从外部文件读入一个数值型向量, S提供了scan()函数。如果指定了file参数(也是第一参数), 则从指定文件读入, 缺省情况下读入一个数值向量, 文件中各数据以空白分隔, 读到文件尾为止。例如：

```
> cat(1:12, '\n', file='c:/work/result.txt')
> x <- scan('c:/work/result.txt')
```

如果文件中是一个用空白分隔的矩阵(或数组), 我们可以先用scan()把它读入到一个向量然后用matrix()函数(或array()函数)转换。如：

```
> y <- matrix(scan('c:/work/result.txt'),
       ncol=3, byrow=T)
```

实际上, scan()也能够读入一个多列的表格, 只要用what参数指定一个列表, 则列表每项的类型为需要读取的类型。用skip参数可以跳过文件的开始若干行不读。用sep参数可以指定数据间的分隔符。详见帮助。

scan()不指定读取文件名时是交互读入, 读入时用一个空行结束。

如果要读取一个数据框, S提供了一个read.table()函数。它只要给出一个文件名, 就可以把文件中用空白分隔的表格数据每行读入为数据框的一行。比如, 文件`c:\work\d.txt`中内容如下：

```
Zhou     15 3
"Li Ming" 9 李明
Zhang 10.2 Wang
```

用read.table读入：

```
> x <- read.table('c:/work/d.txt',
+ colClasses=c('character', 'numeric', 'character'))
> x
       V1   V2   V3
1    Zhou 15.0    3
2 Li Ming  9.0 李明
3   Zhang 10.2 Wang
```

读入结果为数据框。函数自动为数据框变量指定“V1”、“V2”这样的变量名, 指定“1”、“2”这样的行名。可以用col.names参数指定一个字符型向量作为数据框的变量名, 用row.names参数指定一个字符型向量作为数据框的行名。函数可以自动识别表列是数值型还是字符型, 并在缺省情况下把字符型数据转换为因子, 这里的第一列和第三列我们并不想转换, 所以人为地加上了一个colClasses参数说明每一列的数据类型。

read.table()可以读入带有表头的文件, 只要加上header=TRUE参数即可。可以用sep参数指定表行各项的分隔符。例如, 为了读入如下带有表头的逗号分隔文件`c:\work\d.csv`:

```
Name,score, cn
Zhou,15,3
Li Ming, 9, 李明
Zhang, 10.2, Wang
```

使用如下语句:

```
> x <- read.table('c:/work/d.csv', header=T, sep=',',
+ colClasses=c('character', 'numeric', 'character'))
> x
           Name score    cn
1          Zhou  15.0     3
2       Li Ming   9.0  李明
3         Zhang  10.2  Wang
```

其它一些用法见帮助。

6.8 程序控制结构

S是一个表达式语言, 其任何一个语句都可以看成是一个表达式。表达式之间以分号分隔或用换行分隔。表达式可以续行, 只要前一行不是完整表达式(比如末尾是加减乘除等运算符, 或有未配对的括号)则下一行为上一行的继续。

若干个表达式可以放在一起组成一个复合表达式, 作为一个表达式使用。组合用大括号表示, 如：

```
> {
> x <- 15
> x
> }
```

S语言也提供了其它高级程序语言共有的分支、循环等程序控制结构。

6.8.1 分支结构

分支结构包括if结构：

if (条件) 表达式1

或

if (条件) 表达式1 else 表达式2

其中的“条件”为一个标量的真或假值, 表达式可以是用大括号包围的复合表达式。有else子句时一般写成：

```
if(条件) {
  表达式组.....
} else {
  表达式组.....
}
```

这样的写法可以使else不至于脱离前面的if。

例如, 如果变量lambda为缺失值就给它赋一个缺省值, 可用：

```
if(is.na(lambda)) lambda <- 0.5;
```

又比如要计算向量x的重对数, 这只有在元素都为正且对数都为正时才能做到, 因此需要先检查：

```
if(all(x>0) && all(log(x))>0) {
  y <- log(log(x));
  print(cbind(x,y));
} else {
  cat('Unable to comply\n');
}
```

注意“&&”表示“与”, 它是一个短路运算符, 即第一个条件为假时就不计算第二个条件, 如果不这样此例中计算对数就可以有无效值。在条件中也可以用“||”(两个连续的竖线符号)表示“或”, 它也是短路运算符, 当第一个条件为真时就不再计算第二个条件。

注：对于计算重对数的问题只要判断all(x>1)也可以, 例子是为了说明短路与。

在用S编程序时一定要时刻牢记S是一个向量语言, 几乎所有操作都是对向量进行的。而S中的if语句却是一个少见的例外, 它的判断条件是标量的真值或假值。比如, 我们要定义一个分段函数f(x), 当x为正时返回1, 否则返回0, 马上可以想到用if语句实现如下：

```
if(x>0) 1 else 0
```

当x是标量时这个定义是有效的, 但是当自变量x是一个向量时, 比较的结果也是一个向量, 这时条件无法使用。所以, 这个分段函数应该这样编程：

```
y <- numeric(length(x))
y[x>0] <- 1
y[x<=0] <- 0
y
```

有多个if语句时else与最近的一个配对。可以使用“if ··· else if ··· else if ··· else ···” 的多重判断结构表示多分支。多分支也可以使用switch()函数。

6.8.2 循环结构

循环结构中常用的是for循环, 是对一个向量或列表的逐次处理, 格式为“for(*name* in *values*) 表达式”, 如：

```
s <- 0
for(i in seq(along=x)){
  cat('x(', i, ') = ', x[i], '\n', sep='');
  s <- s+x[i];
}
```

这个例子我们需要使用下标的值, 所以用seq(along=x)生成了x的下标向量。如果不需要下标的值, 可以直接如此使用：

```
s <- 0
for(xi in x){
  cat(xi, '\n')
  s <- s + xi
}
```

当然, 如果只是要求各元素的和, 只要调用sum(x)即可。从这里我们也可以看出, 显式的循环经常是可以避免的, 利用函数对每个元素计算值、使用sum等统计函数及apply、lapply、sapply、tapply等函数往往可以代替循环。因为循环在S中是很慢的(S-PLUS和R都是解释语言), 所以应尽可能避免使用显式循环。

我们再举一个例子。比如, 我们要计算同生日的概率。假设一共有365个生日(只考虑月、日), 而且各生日的概率是相等的(这里忽略了闰年的情况以及可能存在的出生日期分布的不均匀)。设一个班有n个人, 当n大于等于365时至少两个人生日相同是必然时间。当n小于365时, 我们可以计算Pr{至少有两人同生日}= 1 - Pr{n个人生日彼此不同}, 这时, n个人的生日可取值数为365^n, 而n个人彼此不同的可能数为365中取n个的排列数, 彼此不同的概率为$365 \times 364 \times \cdots \times (365 - (n-1)) = 365!/(365-n)!$。因此, 为了计算n=1, 2, ..., 364的情况下的同生日概率, 可以用如下循环实现：

```
x <- numeric(364)
for(n in 1:364){
```

```
  x[n] <- 1
  for(j in 0:(n-1)){
    x[n] <- x[n] * (365-j)/365
  }
  x[n] <- 1 - x[n]
}
```

这段程序运行了31秒。我们可以尽量用向量运算来实现, 速度要快得多：

```
x <- numeric(364)
for(n in 1:364){
  x[n] <- 1 - prod((365:(365-n+1))/365)
}
```

这段程序只用了不到1秒。注意不能直接去计算365!, 这会超出数值表示范围。

另外要注意使用for(i in 1:n)格式的计数循环时要避免一个常见错误, 即当n为零或负数时1:n是一个从大到小的循环, 而我们经常需要的是当n为零或负数时就不进入循环。为达到这一点, 可以在循环外层判断循环结束值是否小于开始值。

while循环是在开始处判断循环条件的当型循环, 如：

```
while(b-a>eps){
  c <- (a+b)/2;
  if(f(c)>0) b <- c else a <- c
}
```

是一段二分法解方程的程序, 要求函数f(x)是单调上升的。可以很容易地把这个程序修改成允许f(x)单调下降。

还可以使用

repeat 表达式

循环, 在循环体内用break跳出。

在一个循环体内用next表达式可以进入下一轮循环。

分支和循环结构主要用于定义函数。

6.9 S程序设计

对于复杂一些的计算问题我们应该编写成函数。这样做的好处是编写一次可以重复使用, 并且可以很容易地修改, 另外的好处是函数内的变量名是局部的, 运行函数不会使函数内的局部变量被保存到当前的工作空间, 可以避免在交互状态下直接赋值定义很多变量使得工作空间杂乱无章。

6.9.1 工作空间管理

前面我们已经提到, S在运行时保持一个变量搜索路径表, 要读取某变量时依次在此路径表中查找, 返回找到的第一个；给变量赋值时在搜索路径的第一个位置赋值。但是, 在函数内部, 搜索路径表第一个位置是局部变量名空间, 所以变量赋值是局部赋值, 被赋值的变量只在函数运行期间有效。

用ls()函数可以查看当前工作空间保存的变量和函数, 用rm()函数可以剔除不想要的对象。如：

```
> ls()
 [1] "A"     "Ai"    "b"     "cl"    "cl.f"  "fit1"  "g1"    "marks" "ns"
[10] "p1"    "rec"   "tmp.x" "x"     "x1"    "x2"    "x3"    "y"
> rm(x, x1, x2, x3)
```

```
> ls()
 [1] "A"     "Ai"    "b"     "cl"    "cl.f"  "fit1"  "g1"    "marks" "ns"
[10] "p1"    "rec"   "tmp.x" "y"
```

ls()可以指定一个pattern参数, 此参数定义一个匹配模式, 只返回符合模式的对象名。模式格式是UNIX中grep的格式。比如, ls(pattern="tmp[.]")可以返回所有以"tmp."开头的对象名。

rm()可以指定一个名为list的参数给出要删除的对象名, 所以rm(list=ls(pattern="tmp[.]"))可以删除所有以"tmp."开头的对象名。

6.9.2 函数定义

S中函数定义的一般格式为"函数名<- function(参数表) 表达式"。定义函数可以在命令行进行, 例如:

```
> hello <- function(){
+ cat("Hello, world!\n")
+ cat("\n")
+ }
> hello
function(){
cat("Hello, world!\n")
cat("\n")
}
```

函数体为一个复合表达式, 各表达式的之间用换行或分号分开。不带括号调用函数显示函数定义, 而不是调用函数。

在命令行输入函数程序很不方便修改, 所以我们一般是打开一个其他的编辑程序(如Windows的记事本), 输入以上函数定义, 保存文件, 比如保存到了C:\work\hello.r, 我们就可以用

```
> source("c:\\work\\hello.r")
```

运行文件中的程序。实际上, 用source()运行的程序不限于函数定义, 任何S程序都可以用这种方式编好再运行, 效果与在命令行直接输入基本相同, 只不过不自动显示表达式值。

对于一个已有定义的函数, 可以用fix()函数来修改, 如：

```
> fix(hello)
```

将打开一个编辑窗口显示函数的定义, 修改后关闭窗口函数就被修改了。fix()调用的编辑程序缺省为记事本, 可以用“options(editor=”编辑程序名”)”来指定自己喜欢的编辑程序。

函数可以带参数, 可以返回值, 例如：

```
larger <- function(x, y){
  y.is.bigger <- (y>x);
  x[y.is.bigger] <- y[y.is.bigger]
  x
}
```

这个函数输入两个向量(相同长度)x和y, 然后把x中比y对应元素小的元素替换为y中对应元素, 返回y的值。S返回值为函数体的最后一个表达式的值, 不需要使用return()函数。不过, 也可以使用“return(*对象*)”函数从函数体返回调用者。

6.9.3 参数(自变量)

函数可以带虚参数(形式自变量)。S函数调用方式很灵活, 例如, 如下函数：

```
fsub <- function(x, y) x-y
```

有两个虚参数x和y, 我们用它计算100 − 45, 可以调用fsub(100,45), 或fsub(x=100,y=45), 或fsub(y=45, x=100), 或fsub(y=45, 100)。即调用时实参与虚参可以按次序结合, 也可以直接指定虚参名结合。实参先与指定了名字的虚参结合, 没有指定名字的按次序与剩下的虚参结合。

函数在调用时可以不给出所有的实参, 这需要在定义时为虚参指定缺省值。例如上面的函数改为：

```
fsub <- function(x, y=0) x-y
```

则调用时除了可以用以上的方式调用外还可以用fsub(100), fsub(x=100)等方式调用, 只给出没有缺省值的实参。

即使没有给虚参指定缺省值也可以在调用时省略某个虚参, 然后函数体内可以用missing()函数判断此虚参是否有对应实参, 如：

```
trans <- function(x, scale){
  if(!missing(scale)) x <- scale*x
   • • • • • • • • • • • • • •
}
```

此函数当给了scale的值时对自变量x乘以此值, 否则保持原值。这种用法在其它语言中是极其少见的,

S可以实现这一点是因为S的函数调用在用到参数的值时才去计算这个参数的值(称为“懒惰求值”), 所以可以在调用时缺少某些参数而不被拒绝。

S函数还可以有一个特殊的“...”虚参, 表示所有不能匹配的实参, 调用时如果有需要与其它虚参结合的实参必须用“虚参名=”的格式引入。例如：

```
> f <- function(...){
+   for(x in list(...)){
+     cat(min(x), '\n')
+   }
+ }
> f(c(5,1,2), c(9, 4, 7))
1
4
```

6.9.4 作用域

函数的虚参完全是按值传递的, 改变虚参的值不能改变对应实参的值。例如：

```
> x <- list(1, "abc")
> x
[[1]]
[1] 1

[[2]]
[1] "abc"

> f <- function(x) x[[2]] <- "!!"
> f(x)
> x
[[1]]
[1] 1

[[2]]
```

```
[1] "abc"
```

函数体内的变量也是局部的, 对函数体内的变量赋值当函数结束运行后变量值就删除了, 不影响原来同名变量的值。例如：

```
> x <- 2
> f <- function(){
+ print(x)
+ x <- 20
+ }
> f()
[1] 2
> x
[1] 2
```

这个例子中原来有一个变量x值为2, 函数中为变量x赋值20, 但函数运行完后原来的x值并未变化。但是也要注意, 函数中的显示函数调用时局部变量x还没有赋值, 显示的是全局变量x的值。这是S编程比较容易出问题的地方：你用到了一个局部变量的值, 你没有意识到这个局部变量还没有赋值, 而程序却没有出错, 因为这个变量已有全局定义。

6.9.5 程序调试

S-PLUS和R目前还不象其它主流程序设计语言那样具有单步跟踪、设置断点、观察表达式等强劲的调试功能。调试复杂的S程序, 可以用一些通用的程序调试方法, 另外S也提供了一些调试用函数。

对任何程序语言, 最基本的调试手段当然是在需要的地方显示变量的值。可以用print()或cat()显示。

例如, 我们为了调试前面定义的larger()函数, 可以显示两个自变量的值及中间变量的值：

```
larger <- function(x, y){
  cat('x = ', x, '\n')
  cat('y = ', y, '\n')
  y.is.bigger <- (y>x);
  cat('y.is.bigger = ', y.is.bigger, '\n')
  x[y.is.bigger] <- y[y.is.bigger]
  x
}
```

S提供了一个browser()函数, 当调用时程序暂停, 可以用ls()列出局部变量, 用户可以查看变量或表达式的值, 还可以修改变量。例如：

```
larger <- function(x, y){
  y.is.bigger <- (y>x);
  browser()
  x[y.is.bigger] <- y[y.is.bigger]
  x
}
```

我们运行此程序：

```
> larger(c(1,5), c(2, 4, 9))
Called from: larger(c(1, 5), c(2, 4, 9))
Browse[1]> x
[1] 1 5
Browse[1]> y
[1] 2 4 9
Browse[1]> y>x
[1]  TRUE FALSE  TRUE
Browse[1]> x[y.is.bigger]
[1]  1 NA
Browse[1]> c
[1] 2 5 9
```

退出R的browser()菜单可用c(在S中用return())。在R的browser()状态下用n命令可以进入单步执行状态, 用n或者回车可以继续, 用c可以退出browser, 用Q可以返回到命令行状态。

R提供了一个debug()函数, debug(f)可以打开对函数f()的调试, 执行到函数f时自动进入单步执行的browser()菜单, 回车就可以单步执行。用undebug(f)关闭调试。

6.9.6 程序设计举例

设计S程序是很容易的, 在初学时我们只要使用我们从一般程序设计中学来的知识并充分利用S中现成的各种算法及绘图函数就可以了。但是, 如果要用S编制计算量较大的程序, 或者程序需要发表, 就需要注意一些S程序设计的技巧。

用S语言开发算法, 最重要的一点是要记住S是一个向量语言, 计算应该尽量通过向量、矩阵运算来进行, 或者使用S提供的现成的函数, 避免使用显式循环。显式循环会大大降低S的运算速度, 因为S是解释执行的。

例1. 比如, 考虑核回归问题。核回归是非参数回归的一种, 假设变量Y与变量X之间的关系为:

$$Y = f(X) + \varepsilon$$

其中函数f未知。观测到X和Y的一组样本X_i, Y_i, i=1,...,n后, 对f的一种估计为:

$$f(x) = \sum_{i=1}^{n} K\left(\frac{x-X_i}{h}\right) Y_i \Big/ \sum_{i=1}^{n} K\left(\frac{x-X_i}{h}\right)$$

其中K叫做核函数, 一般是一个非负的偶函数, 原点处的函数值最大, 在两侧迅速趋于零。例如正态密度函数, 或所谓双三次函数核：

$$K(x) = \begin{cases} \left(1 - |x|^3\right)^3 & |x| \le 1 \\ 0 \end{cases}$$

函数density()可以进行核回归估计, 这里作为举例我们对这个算法进行编程。先来编制双三次核函数的程序：

```
kernel.dcube <- function(u){
  y <- numeric(length(u))
  y[abs(u)<1] <- (1 - abs(u[abs(u)<1])^3)^3
  y[abs(u)>=1] <- 0
  y
}
```

注意上面分段函数不用if语句而是采用逻辑向量作下标的办法, 这样定义出的函数允许以向量和数组作自变量。

假设我们要画出核估计曲线, 一般我们取一个范围与各X_i范围相同的等间距向量x然后计算估计出的$f(x)$的值。设观测数据自变量保存在向量X中, 因变量保存在向量Y中, 则按我们一般的程序设计思路, 可以写成下面的S程序：

```
kernel.smooth1 <- function(X, Y, kernel=kernel.dcube,
                  h=1, m=100, plot.it=T){
  x <- seq(min(X), max(X), length=m)
  fx <- numeric(m)
  for(j in 1:m){
    # 计算第j个等间距点的回归函数估计值
    fx[j] <- sum(kernel((x[j]-X)/h)*Y) /
```

```
            sum(kernel((x[j]-X)/h))
  }
  if(plot.it){
    plot(X, Y, type="p")
    lines(x, fx)
  }
  cbind(x=x, fx=fx)
}
```

注意上面程序中用了sum函数来避免一重对i的循环。但是, 上面的写法中仍有一重对j的循环, 使得程序运行较慢。如何改写程序把这个循环也取消呢？办法是把计算看成是矩阵运算。首先, 如果x是一个标量, 则f(x)可以写成：

$$f(x) = (K \cdot Y)/(K \cdot \mathbf{1})$$

其中$K = K\left(\frac{x-X}{h}\right)$是一个与$X$长度相等(即长度为$n$)的行向量, $\mathbf{1}$是一列1。现在, x实际是一个长度为m的向量, 对x的每一个元素可以计算一个长度为n的行向量$K\left(\frac{x[j]-X}{h}\right)$, 把这些行向量上下合并为一个矩阵$K(m \times n)$, 则$KY$为长度为$m$的向量, 每一个元素对应于一个x[j]。合并的矩阵可以用S的outer()函数来计算：

```
f <- function(u, v, kernel, h) kernel((u-v)/h)
K <- outer(x, X, f, kernel=kernel, h=h)
# outer()的第一个自变量对应于结果矩阵的行,
# 第二个自变量对应于列
```

分母为了算每一行的和只要对K右乘$\mathbf{1}$, 于是结果的估计值向量为：

```
fx <- (K %*% Y) /
    (K %*% matrix(1,ncol=1,nrow=length(Y))
```

这样修改kernel.smooth1可以得到更精简的函数kernel.smooth2。

核回归中窗宽h的选择是比较难的, 我们写的核回归函数应该允许用户输入h为一个向量, 对向量中每一个窗宽计算一条拟合曲线并画在图中, 结果作为函数返回值的一列。读者可以作为练习实现这个函数, 并模拟X和Y的观测然后画核估计曲线, $f(x)$用sin(x)。对h我们可能必须用循环来处理了。

例2. Daubechies小波函数计算

小波是近年来得到广泛注意的一个数学工具, 可以应用于各种数学分析问题。小波中一个重要的函数叫做尺度函数(scale function), 它满足所谓双尺度方程:

$$\phi(x) = \sqrt{2}\sum_k h_k \phi(2x-k)$$

一种特殊的尺度函数是只在有限区间上非零的, 叫做紧支集的。紧支集尺度函数可以在给定$\{h_k\}$后用以下迭代公式生成:

$$\eta_0(x) = I_{[-0.5,0.5]}(x)$$
$$\eta_{n+1}(x) = \sqrt{2}\sum_{k=0}^{2N-1} h_k \eta_n(2x-k)$$

其中N是正整数, N=2时h_0=0.482962913145, h_1=0.836516303738, h_2=0.224143868042, h_3 =

-0.129409522551。已知$\phi(x)$的支集(不为零的区间)为$[0, 2N-1]$, $\eta_n(x)$的支集包含于$[-0.5, 2N-1]$中。作为例子, 我们来编写计算$\phi(x)$的S程序。

(1)因为$\phi(x)$是函数, 所以我们只能得到x在$\phi(x)$的支集中的一些等间隔点上的函数值。为了用迭代计算这些值, 我们先写出算法的S伪代码(基本程序结构使用S语法但某些复杂操作用自然语言描述):

```
Daubechies.phi <- function(nsample=256, nrep=20){
  # nsample --- 需要计算的点的个数
  # nrep   --- 迭代次数
  x <- 需要计算的x坐标向量
  N <- 2
  h <- c(0.482962913145, 0.836516303738,
         0.224143868042, -0.129409522551)
  y0 <- 向量, 保存eta(x)在n=0时的初值,
        当x在[-0.5, 0.5]内时为1,其它为0
  for(r in 1:nrep){ # 迭代即循环
    y <- 与x长度相同的0向量
    for(k in 1:(2*N-1)){ # 计算公式中的求和
      yk <- eta(.)在自变量取2x-k处的值*sqrt(2),
            用上一步的y0来表达
      y <- y + yk
    } # for k
    y0 <- y  # 把新的eta(.)放进y0中
             # 作为下一步迭代的开始值
  } # for r
  把x和y只保留x在[0, 2*N-1]中的部分
  返回x和y作为(x[i], phi(x[i])
}
```

(2)下面我们逐步细化这个算法, 并根据需要对算法进行必要的修改。设x[i]=a+(i-1)/(nsample-1)*(b-a), 我们发现, 这个算法中关键问题是如何把$2x-k$处的函数值从y0中查到。理想的情况是2*x[i]-k恰好

等于某一个x[j], 这时eta(2*x[i]-k)=eta(x[j])=y0[j]。如果2*x[i]-k在两个x[j]之间, 可以取最近的一个x[j]然后使用y0[j]。经试验, 这样的做法不能保证迭代收敛。所以, 应该调整算法使得每一个2*x[i]-k都恰好落在某个x[j]上面。这要求

$$2a + \frac{2(i-1)}{n-1}(b-a) = a + \frac{j-1}{n-1}(b-a)$$

其中n为采样点数。推导得

$$j = 1 + 2(i-1) - (k+1)\frac{n-1}{b-a}$$

所以只要$n-1$能被$b-a$整除即可。为此, 取$a=-1$, $b=2N-1$, $n=1+(2N)m$, m为整数。但是, 这样取的n个点包含了区间$[-1,0)$, 在最后的结果中是要丢弃的, 所以如果采样至少要nsample个点的话应该保证n*(2N-1)/(2N)大于等于nsample。

(3) 如果必须要求按指定的采样点数给结果的话可以用线性插值来得到所需的结果。线性插值的函数为approx(x, y, n), 其中(x, y)为要插值的散点(两个向量), n是要求的采样点数(从x的最小值到最大值均匀采样)。

(4) 最后的程序如下:

```
# S-examp2.r
# Daubechies 迭代有限支集正交小波基构造

Daubechies.phi <- function(nsample=256, nrep=20,
                           plot.it=T, debugging=F,
                           nsample.exact=F){
  # nsample: 采样点数
```

```
# plot.it: 是否每次绘图
# debuggin: 是否输出调试信息
# nsample.exact: 是否严格要求采样点数.
#                取FALSE时可以适当放大.

# 注意: 所有eta(x)的支撑的并集为[-0.5, 2N-1],
# 但为了迭代时计算的点都落在采样点上所以要把
# 左端点设为-1, 把采样个数适当放大.

N <- 2
a <- -1
b <- 2*N-1  # 采样区间[-1, 2N-1]
ns <- ceiling((nsample*(2*N)/(2*N-1)-1)/(2*N))*(2*N) + 1
  # ns: 新的采样点数
m <- (ns-1)/(2*N)
  # m: phi(x)的采样密度(x轴每单位有多少个点)
h <- c(0.482962913145, 0.836516303738,
       0.224143868042, -0.129409522551) * sqrt(2)

x <- seq(from=-1, to=2*N-1, length=ns)
  # x: phi(x)的采样位置
y0 <- numeric(ns) # 初始函数
cond1 <- (x>=-0.5 & x<=0.5)
y0[cond1] <- 1.0
y0[!cond1] <- 0.0

for(r in 1:nrep){  # 迭代次数
  if(debugging && plot.it){
    plot(x, y0, type="l", main=paste("Iteration", r))
    #locator(n=1)
  }
  y <- numeric(ns)  # 迭代结果
  for(k in 0:(2*N-1)){
    # 首先计算eta(2x-k),
    # 而x对应于a, a+1/m, a+2/m, ..., b=a+(ns-1)/m
    # 共ns个点, 我们只要找出2x-k对应的下标即可.
    # 这些下标里面有些是出界的,
    # 只要找在1到ns内的即可.
    yk <- numeric(ns) # 保存eta(2x-k)
    i <- seq(along=x)
```

```
      j <- 2*i - 1 - (k+1)*m
      cond2 <- (j>=1) & (j<=ns)
      yk[cond2] <- y0[j[cond2]]*h[k+1]
      y <- y + yk
    } # k
    y0 <- y # 新的迭代初值
  }   # r

  xphi <- x[(1+m):ns]
  yphi <- y[(1+m):ns]
  ns <- ns - m
    # 当前phi(x)从0到2N-1采样的个数.
    # 去掉了[-1, 0)的m个点

  if(plot.it){
    plot(xphi, yphi, type="l",
         main=expression(phi(x)), xlab="x", ylab="")
    abline(h=0)
  }

  if(nsample.exact){
    # 必须要求nsample个点.
    # 使用线性插值(approx, approxfun)
    # 或样条插值(spline, splinefun)
    xy1 <- approx(xphi, yphi, n=nsample)
    xphi <- xy1$x
    yphi <- xy1$y
  }

  invisible(list(xphi=xphi, yphi=yphi))
}
```

结果见图6.3。

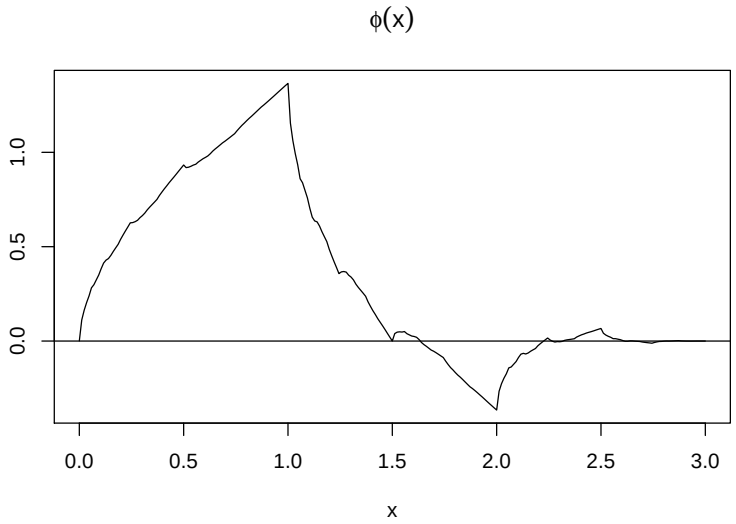


图 6.3: N=2的Daubechies小波尺度函数

6.10 图形

6.10.1 常用图形

S有很强的图形功能, 它可以用简单的函数调用迅速作出数据的各种图形, 当你熟悉了S图形的技术之后也可以指定许多图形选项按自己的要求定制图形。它的另一个特色是同一个绘图函数对不同的数据对象可以作出不同的图形。例如, 用6.1.2读入的cl数据框:

```
> attach(cl)
> plot(Height)
> plot(Sex)
```

第一个plot()绘制身高的散点图(图6.4), 第二个plot()绘制性别频数条形图(图6.5)。

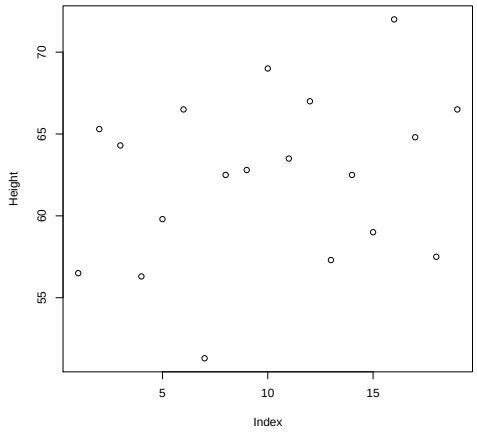


图 6.4: 身高散点图

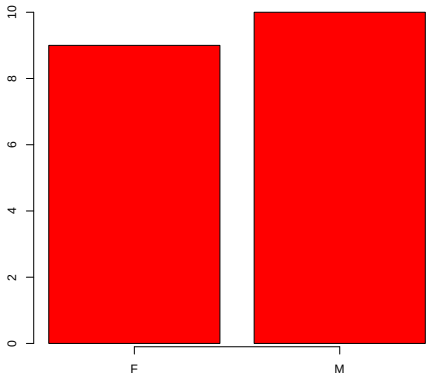


图 6.5: 性别频数条形图

最常用的绘图函数为plot(), 用plot()作两个变量x与y的散点图, 使用如下例的方法：

```
> plot(Height, Weight, main="体重对身高的回归",
+ xlab="身高", ylab="体重")
```

见图6.6。上例也演示了S中如何输入较长的语句：只要语句明显地未完成(比如, 缺右括号), 系统将给出一个加号作为续行提示。如果输入“x <- 1+2”时要拆行, 可以在赋值号后拆, 可以在加号与2之间拆, 但是如果在x后拆则只能显示x的当前值, 如果在1与加号之间拆只能把1赋给x。

为了绘制连线图, 只要在plot()函数中加type=”l”选项, 如：

```
> plot(((-50):50)/25, (((-50):50)/25)^3, type="l")
```

见图6.7。可以绘制变量的茎叶图, 如：

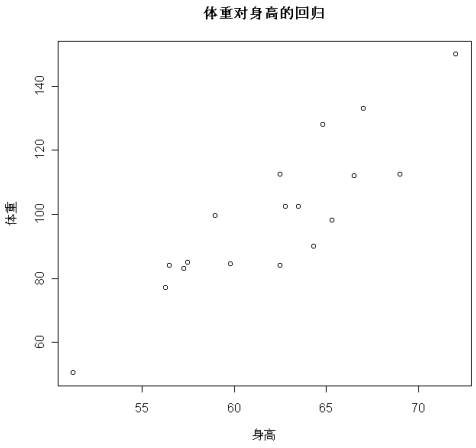


图 6.6: 体重对身高散点图

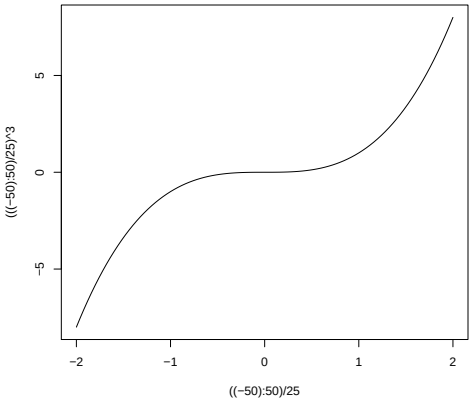


图 6.7: 三次多项式曲线

```
> stem(Height)

  The decimal point is 1 digit(s) to the right of the |

  5 | 1
  5 | 67789
  6 | 033344
  6 | 557779
  7 | 2
```

绘制一个变量的盒形图, 如:

```
> boxplot(Weight)
```

结果见图6.8。可以绘制几个变量并排的盒形图, 比如先计算Weight对Height的回归把拟合结果存入p1, 然后绘制并排盒形图:

```
> fit1 <- lm(Weight ~ Height)
> p1 <- predict(fit1, cl)
```

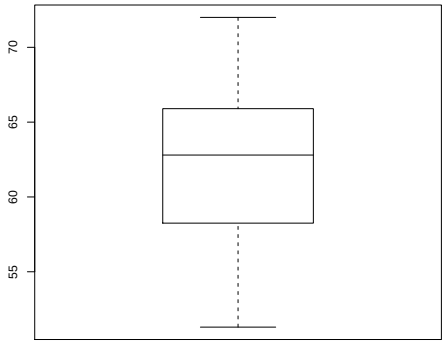


图 6.8: 体重的盒形图

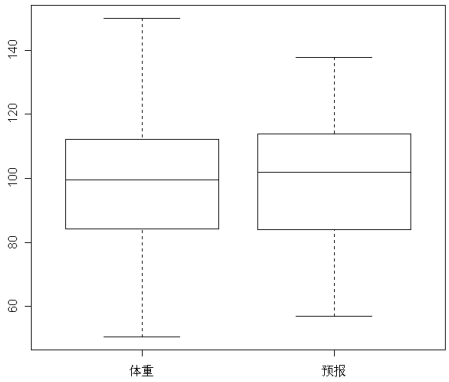


图 6.9: 并排盒形图

```
> boxplot(list("体重"=Weight, "预报"=p1))
```

见图6.9。用hist()函数可以绘制直方图。例如：

```
> hist(Weight)
```

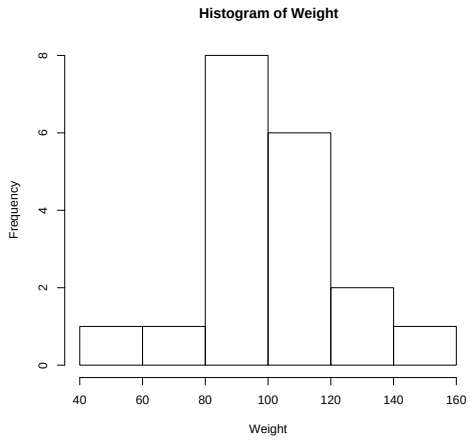


图 6.10: 体重的直方图

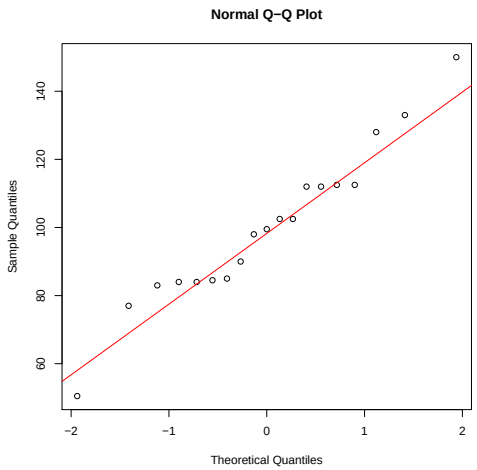


图 6.11: 体重的正态概率图

见图6.11。用qqnorm()函数绘制正态概率图, 如：

```
> qqnorm(Weight)
> qqline(Weight, col="red")
```

见图6.11。

6.10.2 高级图形函数

S的图形函数分为两类：高级图形函数–直接绘制图形并可自动生成坐标轴等附属图形元素；低级图形函数–可以修改已有的图形或者为绘图规定一些选择项。高级图形函数总是开始一个新图。下面我们介绍常用的高级图形函数, 以及用来修饰这些高级图形函数的常用可选参数。

最常用的是plot()函数。比如, plot(x,y)(其中x, y是向量)对两个变量画散点图。用plot(z)(其中z是一个定义了x变量和y变量的列表, 或者一个两列的矩阵)也可以达到同样目的。如果x是一个时间序列对象(时间序列对象用ts()函数生成), plot(x)绘制时间序列曲线图。如果x是一个普通向量, 则绘制x的值对其下标的散点图。如果x是复数向量则绘制虚部对实部的散点图。如果f是一个因子, 则plot(f)绘制f的条形图(每个因子水平的个数)。如果f是因子, y是同长度的数值向量, 则plot(f,y)对f的每一因子水平绘制y中相应数值的盒形图。如果d是一个只有数值型数据的数据框, 则plot(d)对d的每两个变量之间作图(散点图等)。

如果X是一个数值型矩阵或数据框, 用pairs(X)可以绘制每两列之间的散点图矩阵。这在变量个数不太

多时可以同时看到多个变量的两两关系, 变量太多时则难以绘制。

协同图(coplot)是一种多变量的探索性分析图形。其形式为coplot(y ~ x | z), 其中x和y是数值型向量, z是同长度的因子。对z的每一水平, 绘制相应组的x和y的散点图。如:

```
> attach(cl)
> coplot(Weight ~ Height | Sex)
```

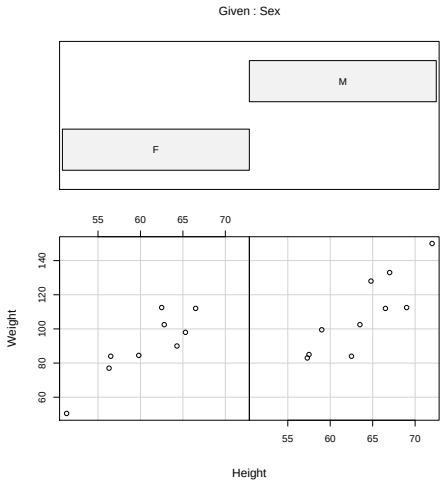


图 6.12: 以性别为条件的体重对身高的散点图

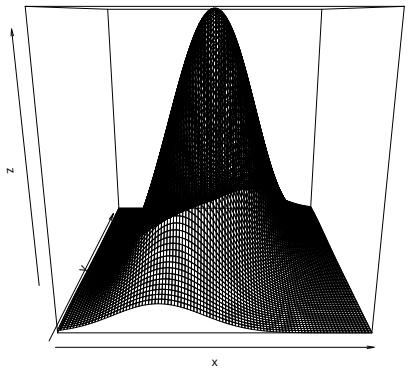


图 6.13: 二维正态密度曲面

产生图6.12, 对不同性别分别绘制了体重对身高的散点图。如果z是一个数值型变量, 则coplot()先对z的取值分组, 然后对z的每一组取值分别绘图。甚至可以用如coplot(y~ x | x1+x2)表示对x1和x2的每一水平组合绘图。coplot()和pairs()函数缺省绘制散点图, 但可以用一个panel=参数指定其它的低级绘图函数, 如lines, panel.smooth等。

qqnorm(x), qqline(x), qqplot(x,y)作分位数-分位数图。qqnorm(x)对向量x作正态概率(纵轴为次序统计量值, 横轴为对应该次序统计量的标准正态分布分位数值)。qqline(x)在qqnorm(x)图上面画一条拟合曲线。qqplot(x,y)把x和y的次序统计量分别画在x轴和y轴以比较两个变量的分布。

hist(x)作向量x的直方图。缺省时自动确定分组, 也可以用nclass=参数指定分组个数, 或者用breaks=参数指定一个分组点向量。如果指定了prob=T则纵轴显示密度估计。

S也可以作三维图、等值线图和等值色图, 函数为persp(), contour() 和image(), 如:

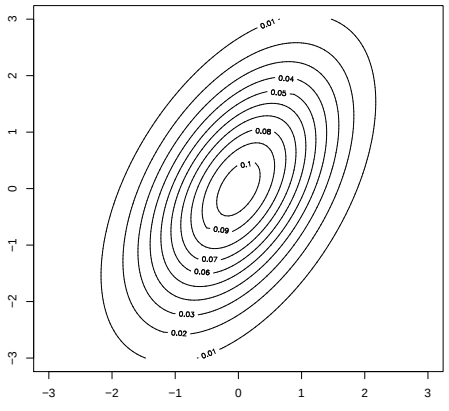


图 6.14: 等值线图

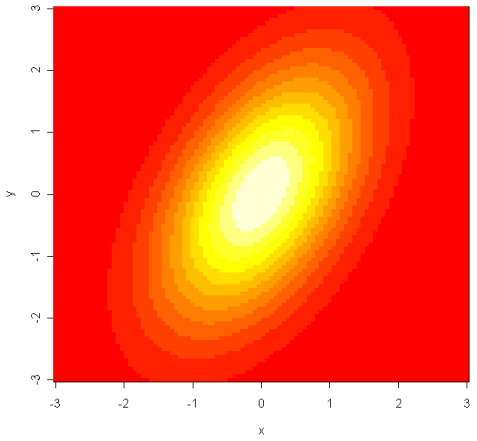


图 6.15: 等值色图

```
x <- seq(-3,3, length=100)
y <- x
f <- function(x,y,ssq1=1, ssq2=2, rho=0.5){
     det1 <- ssq1*ssq2*(1 - rho^2)
     s1 <- sqrt(ssq1)
```

```
      s2 <- sqrt(ssq2)
      1/(2*pi*det1) * exp(-0.5 / det1 * (
        ssq2*x^2 + ssq1*y^2 - 2*rho*s1*s2*x*y))
}
z <- outer(x, y, f)
persp(x, y, z)
contour(x, y, z)
image(x, y, z)
```

见图6.13, 图6.14, 图6.15。

6.10.3 高级图形函数的常用选项

高级图形函数有一些共同的选项, 作为函数的可选参数(自变量)。例如：

```
> plot(x, main="Graph of x")
```

其中的main就是一个可选参数, 用来指定图形的标题。没有此选项时图形就没有标题。表6.1列出了这样的选项。

6.10.4 低级图形函数

高级图形函数可以迅速简便地绘制常见类型的图形, 但是, 某些情况下你可能希望绘制一些有特殊要求的图形。比如, 你希望坐标轴按照自己的设计绘制, 在已有的图上增加另一组数据, 在图中加入一行文本注释, 绘出多个曲线代表的数据的标签, 等等。低级图形函数让你在已有的图的基础上进行添加。

表6.2列出了常用的低级图形函数。

表 6.1: 高级图形选项

add=T	使函数向低级图形函数那样不是开始一个新图形而是在原图基础上添加。
axes=F	暂不画坐标轴, 随后可以用axis()函数更精确地规定坐标轴的画法。缺省值是axes=T, 即有坐标轴。
log="x" log="y" log="xy"	把x轴, y轴或两个坐标轴用对数刻度绘制。
type= type="p" type="l" type="b" type="o" type="h" type="s" type="S" type="n"	规定绘图方式： 绘点 画线 绘点并在中间用线连接 绘点并画线穿过各点 从点到横轴画垂线 阶梯函数；左连续 阶梯函数；右连续 不画任何点、线, 但仍画坐标轴并建立坐标系, 适用于后面用低级图形函数作图。
xlab="字符串" ylab="字符串"	定义x轴和y轴的标签。缺省时使用对象名。
main="字符串"	图形的标题。
sub="字符串"	图形的小标题, 用较小字体画在x轴下方。

低级图形函数一般需要指定位置信息, 其中的坐标指的是所谓用户坐标, 即前面的高级图形函数所建立的坐标系中的坐标。坐标可以用两个向量x和y给出, 也可以由一个两列的矩阵给出。如果交互作图可以用下面介绍的locator()函数来交互地从图形中直接输入坐标位置。

表 6.2: 低级图形函数

points(x,y) lines(x,y)	在当前图形上叠加一组点或线。可以使用plot()的type=参数来指定绘制方法, 缺省时points() 画点,lines()画线。
text(x,y, labels, ...)	在由坐标x和y给出的位置标出由labels指定的字符串。labels可以是数值型或字符型的向量,labels[i]在x[i],y[i]处标出。
abline(a, b) abline(h=y) abline(v=x) abline(lm.obj)	在当前图形上画一条直线。两个参数a, b分布给出截距和斜率。指定h=参数时绘制水平线,指定v=参数时绘制垂直线。以一个最小二乘拟合结果lm.obj作为参数时由lm.obj的$coefficients 成员给出直线的截距和斜率。
polygon(x, y, ...)	按向量x给出的横坐标和向量y给出的纵坐标确定顶点绘制多边形。可以用col=参数指定一个颜色填充多边形内部。
legend(x, y, legend, ...)	legend函数用来在当前图形的指定坐标位置绘制图例。图例的说明文字由向量legend提供。至少下面的v值要给出以确定要对什么图例进行说明, v是长度与legend相同的向量。
legend(, angle=v)	angle参数指定几种阴影斜角。
legend(, density=v)	density参数指定几种阴影密度。
legend(, fill=v)	fill参数指定几种填充颜色。
legend(, col=v)	col参数指定几种颜色。
legend(, lty=v)	lty参数指定几种线型。
legend(, pch=v)	pch参数指定几种散点符号。为字符型向量。
legend(, marks=v)	marks参数也指定几种散点符号,但使用散点符号数值代号,为数值型向量。
title(main, sub)	绘制由main指定的标题和由sub指定的小标题。
axis(side, ...)	绘制一条坐标轴。这之前的绘图函数必须已经用axes=F选项抑制了自动的坐标轴。参数side 指定在哪一边绘制坐标轴,取值为1到4,1为下边,然后逆时针数。可以用at=参数指定刻度位置,用labels参数指定刻度处的标签。

6.10.5 交互图形函数

S的低级图形函数可以在已有图形的基础上添加新内容, 另外, S还提供了两个函数locator和identify可以让用户通过在图中用鼠标点击来确定位置。

函数locator(n, type)运行时会停下来等待用户在图中点击, 然后返回图形中鼠标点击的位置的坐标。等待点击时用鼠标右键点击可以选择停止等待, 立即返回。参数n指定点击多少次后自动停止, 缺省为500次；参数type如果使用则可指定绘点类型, 与plot()函数中的type参数用法相同, 在鼠标点击处绘点(线、垂线, 等等)。locator()的返回值是一个列表, 有两个变量(元素)x和y, 分别保存点击位置的横坐标和纵坐标。

例如, 为了在已经绘制的曲线图中找一个空地方标上一行文本, 只要使用如下程序：

```
> text(locator(1), "Normal density", adj=0)
```

text()函数的adj参数用一个数字表示文本串相对于给定的坐标的画法, adj=0表示给定坐标为文本串左侧的坐标, adj=1表示给定坐标为文本串右侧的坐标, adj=0.5表示给定坐标为文本串中间的坐标。

函数identify(x, y, labels)在运行时也会停下来等待用户点击, 直到按了鼠标右键, 然后返回用户在图形中用鼠标点击的点的序号, 点击时对点击的点加标签。参数x和y给出要识别的各个点的坐标。labels参数指定点击某个点时要在旁边绘制的文本标签, 缺省时标出此点的序号, 如果只需要返回值而不想画任何标记则可以在调用此函数时加一个plot=F参数。注

意identify()与locator()不同, locator()返回图中任意点击位置的坐标, 而identify()只返回离点击位置最近的点的序号。

例如, 我们在向量x和y中有若干个点的坐标, 运行如下程序：

```
> plot(x, y)
> identify(x, y)
```

这时显示转移到图形窗口, 进入等待状态, 用户可以点击图中特别的点, 该点的序号就会在旁边标出。为了结束, 只要单击右键并选择停止。返回结果为你点击的各个点的序号。

6.10.6 图形参数的使用

前面我们已经看到了如何用main=, xlab=等参数来规定高级图形函数的一些设置。在实际绘图, 特别是绘制用于演示或出版的图形时, S用缺省设置绘制的图形往往不能满足我们的要求。但是, S提供了一系列所谓图形参数, 通过使用图形参数可以修改图形显示的所有各方面的设置。图形参数包括关于线型、颜色、图形排列、文本对齐方式等各种设置。每个图形参数有一个名字, 比如col代表颜色, col="red"表示红色。每个图形设备有一套单独的图形参数。

设置图形参数分为两种：永久设置与临时设置。永久设置使用par()函数进行设置, 设置后在退出前一直保持有效；临时设置则是在图形函数中加入图形参数, 如上面的例子：

```
> text(locator(1), "Normal density", adj=0)
```

中的adj参数。

par()函数用来访问或修改当前图形设备的图形参数。如果不带参数调用, 如：

```
> par()
$adj
[1] 0.5

$ann
[1] TRUE

. . . . . .

$ylog
[1] FALSE
```

结果为一个列表, 列表的各元素名为图形参数的名字, 元素值为相应图形参数的取值。

如果调用时指定一个图形参数名的向量作为参数, 则只返回被指定的图形参数的列表：

```
> par(c("col", "lty"))
$col
[1] "black"

$lty
[1] "solid"
```

调用时指定名字为图形参数名的有名参数, 则修改指定的图形参数, 并返回原值的列表：

```
> oldpar <- par(col=4, lty=2)
> oldpar

$col
[1] "black"

$lty
[1] "solid"
```

因为用par()修改图形参数是保持到退出以前都有效的, 而且即使是在函数内此修改仍是全局的, 所以我们可以利用如下的惯用法, 在完成任务后恢复原来的图形参数：

```
> oldpar <- par(col=4, lty=2)
• • • • • • (需要修改图形参数的绘图任务)
> par(oldpar) # 恢复原始的图形参数
```

除了象上面那样用par()函数永久修改图形参数, 我们还可以在几乎任何图形函数中指定图形参数作为有名参数, 这样的修改是临时的, 只对此函数起作用。例如：

```
> plot(x, y, pch="+")
```

就用图形参数pch指定了绘散点的符号为加号。这个设定只对这一张图有效, 对以后的图形没有影响。

6.10.7 图形参数详解

鉴于绘制有特殊需要的图形是S的一个强项, 而使用图形参数是完成此类任务的重要手段, 我们在这里较

详细地介绍S的各种图形参数。这些图形参数可以大体上分为以下的几个大类, 我们将分别介绍：

- 图形元素控制
- 坐标轴与坐标刻度
- 图形边空
- 一页多图

一、图形元素

图形由点、线、文本、多边形等元素构成。下列的图形参数用来控制图形元素的绘制细节：

- pch="+" 指定用于绘制散点的符号。绘制的点往往略高于或低于指定的坐标位置,只有pch="."没有这个问题。
- pch=4 如果pch的值为从0到18之间的一个数字,将使用特殊的绘点符号。下例可以显示所有特殊绘点符号：

```
> plot(c(0, 100), c(0, 100), type="n",
+      axes=F, xlab='', ylab='')
> legend(10,90, as.character(0:9), pch=0:9)
> legend(50,90, as.character(10:18), pch=10:18)
```

- lty=2 指定画线用的线型。缺省值lty=1是实线。从2开始是各种虚线。
- lwd=2 指定线粗细,以标准线粗细为单位。这个参数影响数据曲线的线宽以及坐标轴的线宽。下例绘制正弦曲线图：

```
> oldpar <- par(lwd=2)
> x <- (0:100)/100*2*pi
> plot(x, sin(x), type="l", axes=F)
> abline(h=0)
> abline(v=0)
> par(oldpar)
```

- col=2　指定颜色,可应用于绘点、线、文本、填充区域、图象。颜色值也可以用象"red","blue"这样的颜色名指定。
- font=2 用来指定字体的整数。一般font=1是正体,2是**黑体**, 3是*斜体*,4是***黑斜体***。
- font.axis, font.lab, font.main, font.sub　分别用来指定坐标刻度、坐标轴标签、标题、小标题所用的字体。
- adj=-0.1　指定文本相对于给定坐标的对齐方式。取0表示左对齐,取1表示右对齐,取0.5表示居中。此参数的值实际代表的是出现在给定坐标左边的文本的比例, 所以adj=-0.1的效果是文本出现在给定坐标位置的右边并空出相当于文本10%长度的距离。
- cex=1.5　指定字符放大倍数。

二、坐标轴与坐标刻度

许多高级图形带有坐标轴, 还可以先不画坐标轴然后用axis()单独加。函数box()用来画坐标区域四周的框线。

坐标轴包括三个部件：轴线(用lty可以控制线型), 刻度线, 刻度标签。它们可以用如下的图形参数来控制：

- lab=c(5, 7, 12)　第一个数为x轴希望画几个刻度线,第二个数为y轴希望画几个刻度线,这两个数是建议性的；第三个数是坐标刻度标签的宽度为多少个字符,包括小数点,这个数太小会使刻度标签四舍五入成一样的值。
- las=1　坐标刻度标签的方向。0表示总是平行于坐标轴,1表示总是水平,2表示总是垂直于坐标轴。
- mgp=c(3,1,0)　坐标轴各部件的位置。第一个元素为坐标轴位置到坐标轴标签的距离,以文本行高为单位。第二个元素为坐标轴位置到坐标刻度标签的距离。第三个元素为坐标轴位置到实际画的坐标轴的距离,通常是0。
- tck=0.01　坐标轴刻度线长度,单位是绘图区域大小,值为占绘图区域的比例。tck小于0.5时x轴和y 轴的刻度线将统一到相同的长度。取1时即画格子线。取负值时刻度线画在绘图区域的外面。
- xaxs="s", yaxs="d"　控制x轴和y轴的画轴方法。

  取值为"s"（即standard）或"e"（即extended）的时候数据范围控制在最小刻度和最大刻度之间。取"e"时如果有数据点十分靠近边缘轴的范

围会略微扩大。这种画轴方式有时会在轴的一边留下太大的空白。

取值为"i"（即internal）或"r"（此为缺省）使得刻度线都落在数据范围内部,而"r"方式所留的边空较小。

取值设为"d"时会锁定此坐标轴,后续的图形都使用与它完全相同的坐标轴, 这在要生成一系列可比较的图形的时候是有用的。要解除锁定需要把这个图形参数设为其它值。

三、图形边空

S中一个单独的图由绘图区域(绘图的点、线等画在这个区域中)和包围绘图区域的边空组成, 边空中可以包含坐标轴标签、坐标轴刻度标签、标题、小标题等, 绘图区域一般被坐标轴包围。见图6.16。

边空的大小由mai参数或mar参数控制, 它们都是四个元素的向量, 分别规定下方、左方、上方、右方的边空大小, 其中mai取值的单位是英寸, 而mai的取值单位是文本行高度。例如：

```
> par(mai=c(1, 0.5, 0.5, 0))
> par(mar=c(4, 2, 2, 1))
```

这两个图形参数不是独立的, 设定一个会影响另一个。S缺省的图形边空常常太大, 以至于有时图形窗口较小时边空占了整个图形的很大一部分。通常我们可以取消右边空, 并且在不用标题时可以大大缩小上边空。例如下例可以生成十分紧凑的图形：

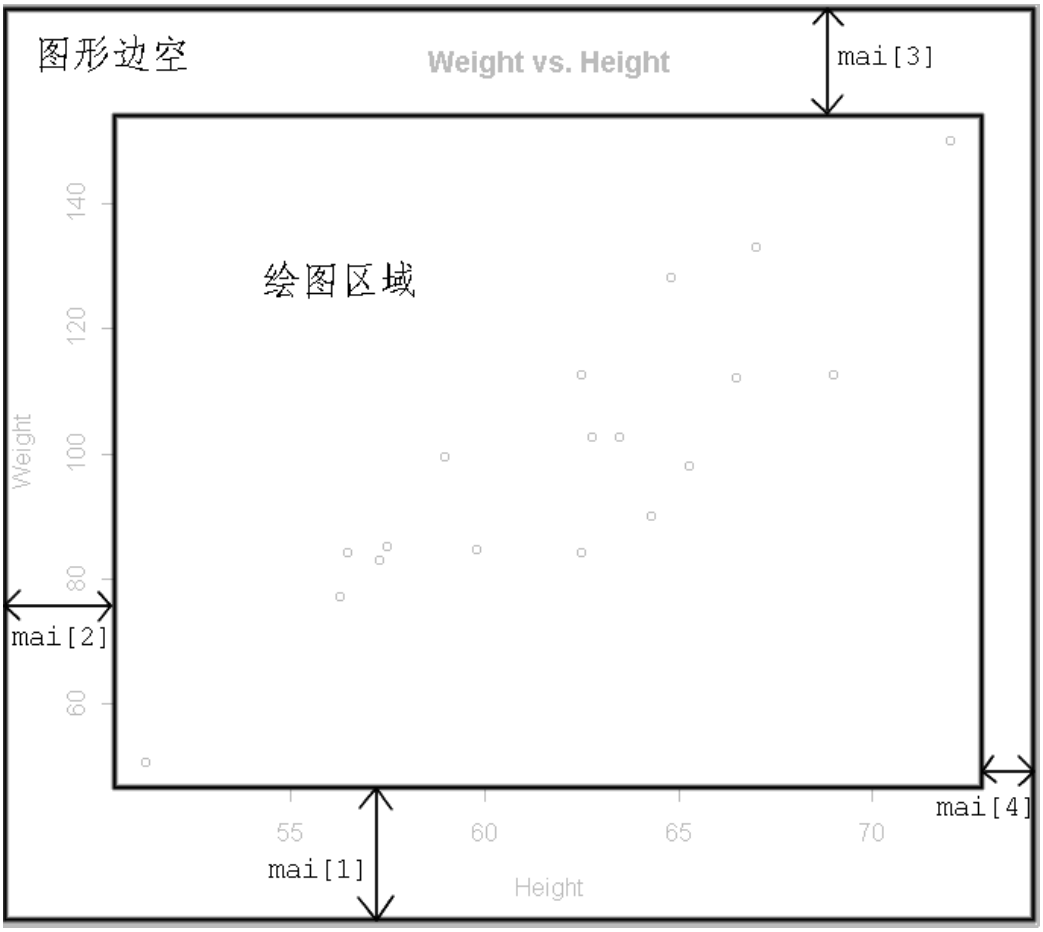


图 6.16: 图形边空

```
> oldpar <- par(mar=c(2,2,1,0.2))
> plot(x,y)
```

在一个页面上画多个图时边空自动减半, 但我们往往还需要进一步减小边空才能使多个图有意义。

四、一页多图

R可以在同一页面开若干个按行、列排列的窗格, 在每个窗格中可以作一幅图。每个图有自己的边空, 而所有图的外面可以包一个“外边空”, 见图6.17。

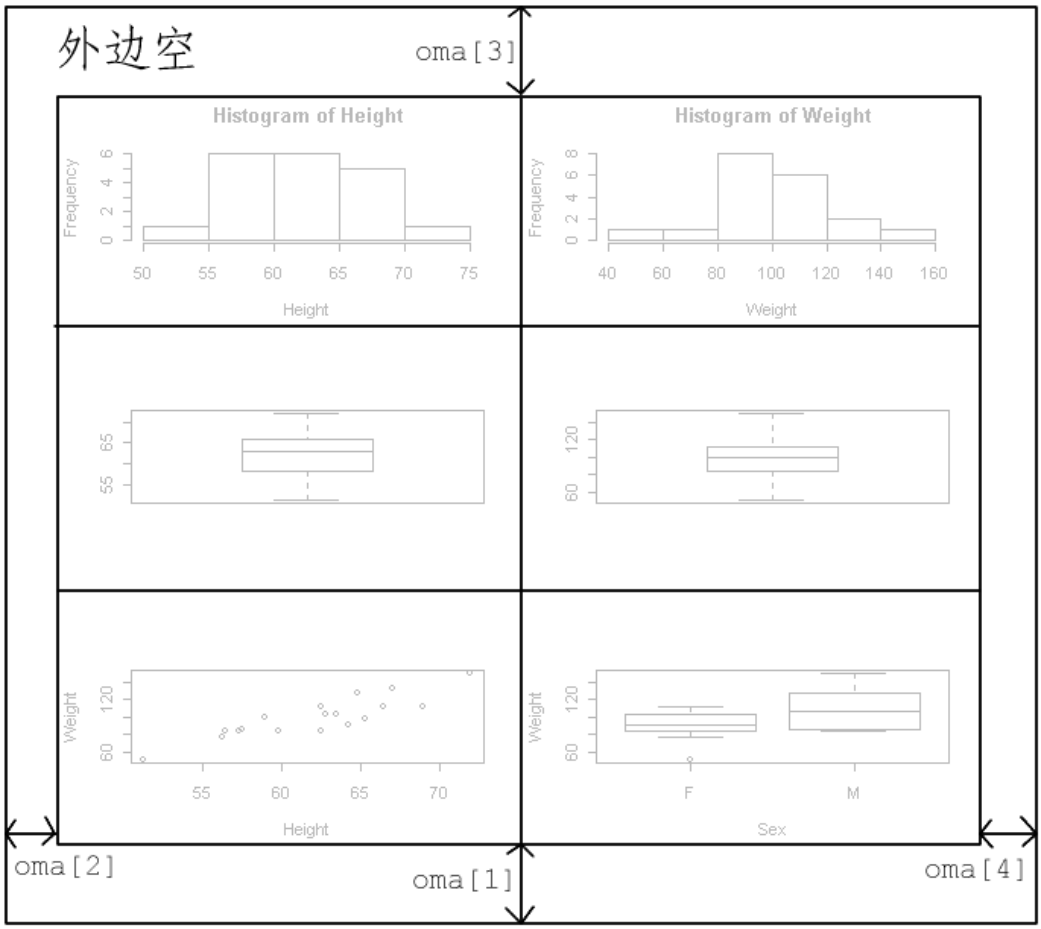


图 6.17: 一页多图

一页多图用mfrow参数或mfcol参数规定, 如：

```
> par(mfrow=c(3,2))
```

表示同一页有三行两列共六个图, 而且次序为按行填放。类似地,

```
> par(mfcol=c(3,2))
```

规定相同的窗格结构, 但是次序为按列填放, 即先填满第一列的三个再填第二列。要取消一页多图只要再运行

```
> par(mfrow=c(1,1))
```

即可。

缺省时无外边空。为了规定外边空大小, 可以在par()中用omi参数或oma参数。omi参数使用英寸为单位, oma参数以文本行高为单位, 两个参数均为四个元素的向量, 分别给出下、左、上、右方的边空大小。如:

```
> par(oma=c(2,0,3,0))
```

函数mtext用来在外边空加文字标注。其用法为

```
    mtext(text, side = 3, line = 0, outer = FALSE)
```

其中text为要加的文本内容, side表示在哪一边写(1为下, 2为左, 3为上, 4为右), line表示边空从里向外数的第几行, 最里面的一行是第0号, outer=TRUE时使用外边空, 否则会使用当前图的边空。例如:

```
> par(mfrow=c(2,2), oma=c(0,0,3,0), mar=c(2,1,1,0.1))
> plot(x);plot(y);boxplot(list(x=x,y=y));plot(x,y)
> mtext("Simulation Data", outer=T, cex=1.5)
```

在多图环境中还可以用mfg参数来直接跳到某一个窗格, 比如

```
> par(mfg=c(2,2,3,2))
```

表示在三行两列的多图环境中直接跳到第二行第二列位置。mfg参数的后两个表示多图环境的行、列数, 前两个表示要跳到的位置。

可以不使用多图环境而直接在页面中的任意位置产生一个窗格来绘图, 参数为fig, 如：

```
> par(fig=c(4,9,1,4)/10)
```

此参数为一个向量, 分别给出窗格的左、右、下、上边缘的位置, 取值为占全页面的比例, 比如上面的例子在页面的右下方开一个窗格作图。

6.10.8 图形设备

S作图支持各种图形设备, 其中常用的是显示器和PostScript打印机。在一个S运行期间可以有多个图形设备同时存在。在R中, 用

```
> x11()
```

打开图形窗口绘图, 在S-PLUS中, 用

```
> win.graph()
```

打开图形窗口绘图。再次调用这样的函数将打开第二个图形窗口。用

```
> dev.list()
```

可显示以打开的图形设备的列表。

要关闭一个图形设备, 用

```
> dev.off()
```

这可以使得图形得以完成, 例如对于postscript设备关闭设备时可完成打印或存盘。用graphics.off()函数可以关闭所有打开的图形设备。

MS Windows下的R可以把显示窗口中的图形复制到剪贴板或存为各种格式的图形文件, 包括WMF, PostScript, PDF, PNG, BMP, JPEG, 这样我们可以用R生成所需图形然后存为需要的格式。MS Windows下的S-PLUS也具有类似功能。

各版本的R和S-PLUS都支持生成PostScript图形的功能, 生成的图形可以直接用于LaTeX排版。如果用MS Word排版则可把屏幕图形存为WMF等格式。生成PostScript文件的设备可以用如下函数打开：

```
> postscript(file="result1.ps",
+ horizontal=FALSE, width=5, height=3)
```

这时用图形命令生成一个页面的图形, 然后用dev.off()关闭设备, 则可生成文

件result1.ps。postscript()函数中horizotal参数指定是否将图旋转90度使得x轴平行于纸的长边, width和height规定图的宽和高, 单位是英寸(1英寸=2.54厘米)。

在打开了多个图形设备后可以用dev.set()函数来选择当前设备, dev.next()和dev.prev()分别返回下一个和上一个图形设备。比如dev.set(dev.prev())选择上一个图形设备。

6.11 S的对象

S是一种面向对象的语言。一般说来, S的对象包含了若干个元素作为其数据, 这些元素的个数叫做此对象的长度(length), 这些元素的共同的类型叫做此对象的模式(mode)。另外, 对象还可以包含一些特殊数据, 称为属性(attribute), 如列表的每一个成员(元素)都可以有变量名, 这些变量名组成的字符型向量为此列表的names属性。

S的面向对象能力依赖于对类属性的使用。S的对象有的是简单对象, 这样的对象没有类(class)属性, 可以认为其类为缺省类(default)；有的是属于某一类的对象, 这些对象有一个类(class)属性, 同类的对象具有相同的特征, 可以为同类的对象定义针对这一类的特殊操作(如显示、绘图), 在面向对象术语中叫做方法。比如, 向量是简单对象, 它没有类属性；数据框也是对象, 但是数据框有一个类属性class=data.frame。

6.11.1 固有属性：mode和length

S对象都有两个基本的属性(attribute)：类型(mode)属性和长度(length)属性。

S对象可分为单纯的(atomic)和复合的(recursive)两种，单纯对象的所有元素都是同一种基本类型(如数值、字符串)，元素不再是对象，这样的对象的类型(mode)有logical(逻辑型)、numeric(数值型)、complex(复数型)、character(字符型)等等；复合对象的元素可以是不同类型，每一个元素是一个对象，这样的对象最常用的是列表。例如，向量(vector)是单纯对象，它的所有元素都必须是相同类型，数值型向量的所有元素必须为数值型，字符型向量的所有元素必须为字符型；列表(list)是复合对象，类型(mode)为列表(list)，列表的每一个元素(变量)都可以是一个S对象，比如列表元素可以为一个数，一个字符串，一个向量，甚至一个列表。

S对象有一种特别的空值类型，只有一个特殊的NULL值为这种类型，表示没有值(不同于NA，NA是一种特殊值，而NULL根本没有对象值)。

为了判断对象的类型，S定义了许多个类似于is.numeric()这样的函数。比如，is.numeric(x)用来检验对象x是否数值型，返回一个逻辑型标量结果；is.character()检验对象是否字符型，等等。

长度属性表示S对象元素的个数，比如length(2:4)等于3。注意向量允许长度为0，数值型向量长度为零表示为numeric()或numeric(0)，字符型向量长度为零表示为character()或character(0)。

S可以强制进行类型转换，例如

```
> z <- 0:9
> digits <- as.character(z)
> digits
 [1] "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"
 > d <- as.numeric(digits)
> d
 [1] 0 1 2 3 4 5 6 7 8 9
```

第二个赋值把数值型的z转换为字符型的digits。第三个赋值把digits又转换为了数值型的d, 这时d和z是一样的了。S还有许多这样的以as.开头的类型转换函数。

S允许对超出对象长度的下标赋值, 这时对象长度自动伸长以包括此下标, 未赋值的元素取缺失值(NA), 例如：

```
> x <- numeric()
> x[3] <- 100
> x
[1] NA NA 100
```

要缩短对象的长度又怎么办呢？只要给它赋一个子集就可以了。例如：

```
> x <- 1:4
> x <- x[1:2]
> x
[1] 1 2
```

6.11.2 访问对象属性

对象属性是对象包含的数据中除元素以外的特殊数据, 每个属性有一个属性名, 有一个属性

值。S定义了两个函数attributes和attr来访问对象的属性。attributes(object)返回对象object的各特殊属性组成的列表, 其中不包括固有属性mode和length。例如：

```
> x <- c(apple=2.5, orange=2.1)
> attributes(x)
$names
[1] "apple"  "orange"
```

可以用attr(object, name)的形式存取对象object的名为name的属性。例如：

```
> attr(x, "names")
[1] "apple" "orange"
```

也可以把attr()函数写在赋值的左边以改变属性值或定义新的属性, 例如：

```
> attr(x, "names") <- c("apple", "grapes")
> x
 apple grapes
   2.5    2.1
> attr(x, "type") <- "fruit"
> x
 apple grapes
   2.5    2.1
attr(,"type")
[1] "fruit"
> attributes(x)
$names
[1] "apple"  "grapes"

$type
[1] "fruit"
```

这种对一个函数赋值的语法是在其它语言中极为少见的, 而S中则经常使用这样的写法。实际上, attr(x, "names")在这里不应该看成是一个函数值, 而应该看成是用来保存对象x的names属性的变量名。

6.11.3 对象的类

S用类(class)属性来支持面向对象的编程风格。对象的类属性区分对象的类, 对于同一类的对象可以定义一组特殊操作, 这一点和其它面向对象语言类似。面向对象风格的最重要的特点就是数据抽象与封装。所谓数据抽象与封装是指对象的用户要访问或修改对象只能通过对象提供的服务来进行, 用户不能看到对象内部的实现细节。这样用户不会直接修改对象的数据从而保护了数据的完整性, 而且用户只需要知道对象提供了哪些服务, 即使对象内部的实现改变了, 只要接口不变则用户程序不必改变。这样的做法可以提高程序的安全性和可重用性。

常见的面向对象语言一般先定义一个类, 这个类定义了一些数据结构, 然后有一些函数叫做"方法"可以操作这些数据。所谓对象, 是由某个类生成的实例, 其数据结构由所属的类定义, 而实际存储的数据则是属于对象本身的。对象拥有其所属类的所有方法, 方法在调用时操作的是属于这个对象的数据。

S也支持面向对象编程, 但是做法与常见的面向对象语言有很大差别。S对象的类由其类(class)属性指定, 每一个类都可以定义本类的服务, 服务以函数形式定义, 调用格式为"函数名(对象, 其它自变量)"。可见S的类机制是比较松散的, 它不象常见的面向对

象语言那样必须先定义类的所有数据结构与方法, 而是可以随时定义函数作为类对象的服务。另外, S还定义了一系列的所谓“通用函数(general functions)”, 通用函数也是对象提供的服务, 但不同类的对象都可以使用相同的通用函数名字调用, 同一个通用函数可以针对不同类的对象起到相似的作用。用户只需要记忆很少的几个通用函数的名字, 就可以对几乎所有对象调用这些函数。比如, 通用print()函数用来显示对象, 它可以显示向量和矩阵, 但显示方法不同；通用函数plot()函数用来画对象的图形, 对一个向量画图plot()画散点图, 纵轴为各元素值, 横轴为元素下标；对一个时间序列对象画图plot()将画一条时间序列曲线, 并用年月等标记时间轴。

S的每一个通用函数实际是一组函数, 有一个共同的名字, 在调用时根据自变量的类(class)的不同决定调用一组中的哪一个函数。例如, 对向量x调用print(x)实际调用的是print.default(x), 对数据框x调用print(x)则实际调用的是print.data.frame(x)。如果自变量没有类属性, 或者此通用函数没有为此类自变量设计特殊的操作, 通用函数总有一个缺省方法可以调用(如print.default)。通用函数针对某一类的对象的特殊函数的命名为“通用函数名.类名()”。

对某一种类的对象有特殊操作的通用函数可以有很多个, 比如, 对data.frame类的对象定义了特殊操作的通用函数就有：

```
[, [[<-, any, as.matrix,
[<-, model, plot, summary,
```

等等。如果对data.frame类的对象d调用plot(d), 实际

调用的函数是plot.data.frame(d)。要列出所有对某类有特殊操作的通用函数, 可以用

```
> methods(class="data.frame")
[1] "Math.data.frame" "Ops.data.frame"
• • • • • • • • • • • •
[27] "summary.data.frame" "t.data.frame"
[29] "transform.data.frame" "xpdrows.data.frame"
```

其中t.data.frame就是调用t(d)时实际调用的函数。

也可以列出某通用函数的对各类的特殊定义, 例如:

```
> methods(plot)
 [1] "plot.data.frame" "plot.default"    "plot.density"    "plot.factor"
 [5] "plot.formula"    "plot.function"   "plot.histogram"  "plot.lm"
 [9] "plot.mlm"        "plot.mts"        "plot.new"        "plot.POSIXct"
[13] "plot.POSIXlt"    "plot.table"      "plot.ts"         "plot.window"
[17] "plot.xy"
```

比如, plot.factor是对因子对象调用plot()函数是实际调用的函数。

为了暂时去掉一个有类的对象的class属性, 可以使用unclass(object)函数。

6.12 S初等统计

S-PLUS和R中已经集成了常用的统计功能, 一些比较新的统计方法也得到了实现。

调用S函数可以完成基本的统计。这里我们介绍如何用S进行描述统计、探索性数据分析、分布研究、常见的假设检验。

6.12.1 单变量数据分析

对放在向量或数据框中的变量我们可以用summary()函数来计算几个描述统计量。例如:

```
> summary(cl)
     Name           Sex         Age            Height          Weight
 Length:19          F: 9   Min.   :11.00   Min.   :51.30   Min.   : 50.50
 Mode  :character   M:10   1st Qu.:12.00   1st Qu.:58.25   1st Qu.: 84.25
                           Median :13.00   Median :62.80   Median : 99.50
                           Mean   :13.32   Mean   :62.34   Mean   :100.03
                           3rd Qu.:14.50   3rd Qu.:65.90   3rd Qu.:112.25
                           Max.   :16.00   Max.   :72.00   Max.   :150.00
```

对数值型变量计算了平均值, 最大、最小值, 中位数, 四分之一和四分之三分位数；对因子计算了频数统计。

对数值型变量x也可以用mean, sd, var, median, min, max等函数计算其简单统计量。可以用stem(x)绘制x的茎叶图, 用hist(x)画直方图, 用boxplot(x)画盒形图, 用qqnorm(x)和qqline(x)画正态QQ图。

为了估计数值型变量的分布密度, 除了用hist()画直方图外还可以用density()函数进行非参数密度估计。例如:

```
plot(density(Height), main="Height density")
```

见图6.18。还可以设法把密度曲线加到直方图上, 如：

```
h1 <- hist(Height, prob=T, plot=T)$density
h2 <- density(Height)
hist(Height, prob=T, ylim=range(h1, h2$y))
lines(h2)
```

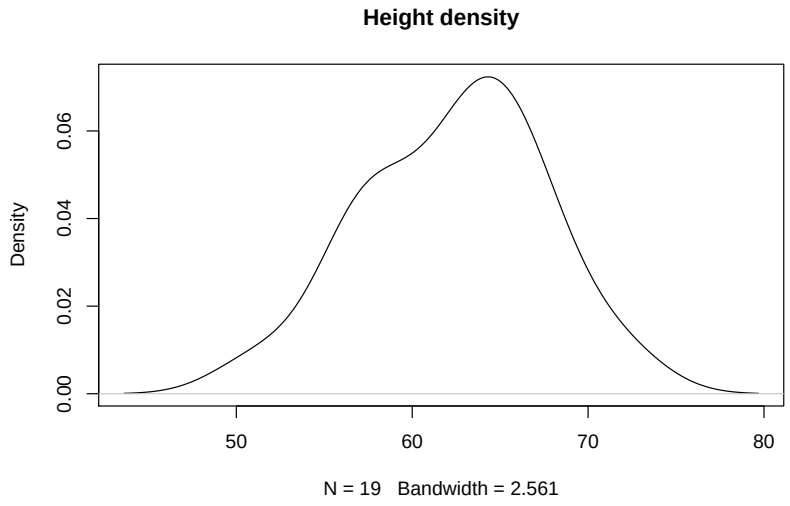


图 6.18: 身高的非参数密度估计

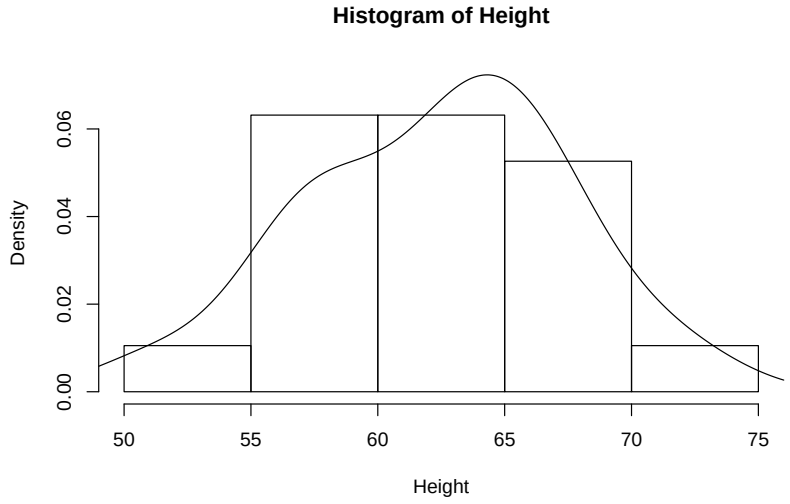


图 6.19: 身高的直方图及密度估计

见图6.19。

对因子(分类变量)f可以用table(f)统计其频数分布,用plot(f)画其频数分布图。如

```
> table(Sex)
Sex
 F  M
 9 10
```

我们可以定义函数来画出常见的探索性数据分析图形。对连续型变量可以画直方图、盒形图、分布密度估计图和正态概率图。对离散型变量只要画其分布频数条形图即可, 分布频数用table函数计算。研究序列相关性可以作时间序列图和自相关函数图。因为这些图经常重复使用, 我们把它们定义为函数,在同一页面画出:

```
eda.shape <- function(x) {
  oldpar <- par(mfcol = c(2, 2),
     mar=c(2,2,0.2,0.2), mgp=c(1.2,0.2,0))
  hist(x, main="", xlab="", ylab="")
  plot(density(x), xlab = "x",
       ylab = "", main="")
  boxplot(x)
  qqnorm(x, main="", xlab="", ylab="")
  qqline(x)
  par(oldpar)
  invisible()
}

eda.ts <- function(x) {
  oldpar <- par(mfrow=c(2,1),
     mar=c(2,2,1,0.2), mgp=c(1.2, 0.2, 0))
  plot.ts(x, main="", xlab="")
  acf(x, main="", xlab="")
  par(oldpar)
```

```
  invisible()
}
```

函数中最后的invisible()表示在命令行调用此函数时不要显示任何返回值。函数density用来作核密度曲线估计。

6.12.2 假设检验

R的ctest程序包(library)提供了常见的假设检验功能。为了调用程序包中的函数和数据, 需要先用library(*包名*)把它调入:

```
> library(ctest)
```

为了查看一个包中的内容可以在它提供的HTML帮助中查看, 也可以用R命令help(*package=包名*)查看。例如

```
> help(package=ctest)
```

为了检验正态性, 只要调用shapiro.test()函数就可以Shapiro-Wilk检验:

```
> shapiro.test(Height)

        Shapiro-Wilk normality test

data:  Height
W = 0.9791, p-value = 0.9312
```

可见身高数据的正态性很好。

函数t.test()可以进行单总体的t检验、两独立总体的t检验和成对t检验。例如,下面程序检验身高均值是否为$\mu = 60$:

```
> t.test(Height, mu=60)

        One Sample t-test

data:  Height
t = 1.9867, df = 18, p-value = 0.06239
alternative hypothesis: true mean is not equal to 60
95 percent confidence interval:
 59.86567 64.80801
sample estimates:
mean of x
 62.33684
```

缺省进行的是双侧检验。为了进行单侧检验可以指定alternative=”greater” 或”less”。如:

```
> t.test(Height, mu=60, alternative="less")

        One Sample t-test

data:  Height
t = 1.9867, df = 18, p-value = 0.9688
alternative hypothesis: true mean is less than 60
95 percent confidence interval:
    -Inf 64.3765
sample estimates:
mean of x
 62.33684
```

对于用分组变量表示分组的两个独立组, 可以用t.test(分析变量~ 分组变量)的调用形式进行独立两组的t检验。如：

```
> t.test(Height ~ Sex)

        Welch Two Sample t-test

data:  Height by Sex
t = -1.4513, df = 16.727, p-value = 0.1652
alternative hypothesis:
true difference in means is not equal to 0
95 percent confidence interval:
 -8.155098  1.512875
sample estimates:
mean in group F mean in group M
       60.58889        63.91000
```

注意上面用的是不需要假定两组方差相等的Welch检验。如果假定两组方差相等可以在函数调用中加参数 var.equal=TRUE。如果两个独立组放在了两个变量X1和X2中则可以直接用 t.test(X1, X2)进行比较。

用wilcox.test(分析变量~ 分组变量) 或wilcox.test (X1, X2) 可以进行独立两组的Wilcoxon 秩和检验。

对于成对观测的变量X, Y,要比较只要用t.test(X, Y, paired=TRUE)。用wilcox.test(X, Y, paired=TRUE)可以进行Wilcoxon符号秩检验。

6.13 S统计模型简介

这一节我们简单介绍S的统计模型。S中实现了几乎所有常见的统计模型, 而且多种模型可以用一种统一的观点表示和处理。这方面S-PLUS较全面, 它实现了许多最新的统计研究成果, R因为是自愿无偿工作所以统计模型部分还相对较欠缺。事实上, 许多统计

学家的研究出的统计算法都以S-PLUS程序发表, 因为S语言是一种特别有利于统计计算编程的语言。

学习这一节需要我们具备线型模型、线型回归、方差分析的基本知识。

6.13.1 统计模型的表示

很多统计模型可以用一个线型模型来表示：

$$y_i = \sum_{j=0}^{p} \beta_j x_{ij} + e_i,\ e_i \sim \text{iid}N(0, \sigma^2),\ i = 1, 2, \ldots, n$$

写成矩阵形式为

$$y = \mathbf{X}\beta + e$$

其中y为因变量, $\mathbf{X}$为模型矩阵或称设计阵, 各列为$x_0, x_1, \ldots, x_p$ 等各自变量, x_0常常是一列1, 定义一个模型截距项。

在S中模型是一种对象, 其表达形式叫做一个公式(formula), 我们先举几个例子来看一看。假定y, x, x0, x1, x2, ...是数值型变量, X是矩阵, A, B, C, ...是因子。

- y ~ x, y ~ 1+x　两个式子都表示y对x的简单一元线型回归。第一个式子带有隐含的截距项,而第二个式子把截距项显式地写了出来。
- y ~ −1+x, y ~ x−1　都表示y对x的通过原点的回归,即不带截距项的回归。
- log(y) ~ x1 + x2　表示log(y)对x1和x2的二元回归,带有隐含的截距项。

- y ∼ poly(x, 2), y ∼ 1 + x + I(x^2)　表示y对x的一元二次多项式回归。第一种形式使用正交多项式,第二种形式直接使用x的各幂次。
- y ∼ X + poly(x,2)　因变量为y的多元回归,模型矩阵包括矩阵X,以及x的二次多项式的各项。
- y ∼ A　一种方式分组的方差分析,指标为y,分组因素为A。
- y ∼ A + x　一种方式分组的协方差分析,指标为y,分组因素为A,带有协变量x。
- y∼A*B, y∼A+B+A:B, y∼B %in% A, y∼A/B 非可加两因素方差分析模型,指标为y,A,B是两个因素。前两个公式表示相同的交叉分类设计,后两个公式表示相同的嵌套分类设计。
- y ∼ (A + B + C)^2, y ∼ A*B*C − A:B:C　表示三因素试验,只考虑两两交互作用而不考虑三个因素间的交互作用。两个公式是等价的。
- y ∼ A * x, y ∼ A / x, y ∼ A / (1 + x) - 1　都表示对因子A的每一水平拟合y对x的线型回归,但三个公式的编码方式不同。最后一种形式对A的每一水平都分别估计截距项和斜率项。
- y ∼ A*B + Error(C)　表示有两个处理因素A和B,误差分层由因素C确定的设计

在S中∼运算符用来定义模型公式。一般的线型模型的公式形式为

因变量∼ 第一项[±] 第二项[±] 第三项[±] ...

其中因变量可以是向量或矩阵, 或者结果为向量或矩阵的表达式。[±]是加号+或者减号−, 表示在模型中加入一项或去掉一项, 第一项前面如果是加号可以省略。

公式中的各项可以取为：

- 一个值为向量或矩阵的表达式, 或1。
- 一个因子
- 一个“公式表达式”, 由“公式运算符”把因子、向量、矩阵连接而成。

每一项定义了要加入模型矩阵或从模型矩阵中删除的若干列。一个1表示一个截距项列, 除非显式地删除总是隐含地包括在模型公式中。

“公式运算符”的定义和Glim、Genstat软件中的定义类似, 不过那里的“.”运算符这里改成了“:”, 因为在S中句点是名字的合法字符。下表列出了各运算符的简要说明。

- $Y \sim M$　Y作为因变量由M解释。
- $M_1 + M_2$　加入M_1和M_2。
- $M_1 - M_2$　加入M_1但去掉M_2指定的项。
- $M_1 : M_2$　M_1和M_2的张量积。如果两项都是因子,则为因子的交互作用。
- M_1 `%in%` M_2　与$M_1 : M_2$类似但模型矩阵编码方式不同。
- $M_1 * M_2$　等于$M_1 + M_2 + M_1 : M_2$。

- $M\hat{}n$　M中所有各项以及所有到n阶为止的交互作用项。
- I(M)　将M隔离,使得M中的运算符按照原来的算术运算符解释而不是按公式运算符解释。表达式M的结果作为公式的一项。

注意在函数调用的括号内的表达式按普通四则运算解释。函数I()可以把一个计算表达式封装起来作为模型的一项使用。

注意S的模型表示只给出了因变量和自变量及自变量间的关系, 这样只确定了线型模型的模型矩阵, 而模型参数向量是隐含的, 并没有的模型公式中体现出来。这种做法适用于线性模型, 但不具有普遍性, 例如非线性模型就不能这样表示。

6.13.2 线性回归模型

拟合普通的线性模型的函数为lm(), 其简单的用法为:

fitted.model <- lm(*formula*, data=*data.frame*)

其中data.frame为各变量所在的数据框, formula为模型公式, fitted.model是线性模型拟合结果对象(其class属性为lm)。例如:

```
mod1 <- lm(Weight ~ Height + Age, data=cl)
```

可以拟合一个y对x1和x2的二元回归(带有隐含的截距项), 数据来自数据框production。拟合的结果存入了对象mod1中。注意不论数据框production是

否以用attach()连接入当前运行环境都可被lm()使用。lm()的基本显示十分简练：

```
> mod1
Call:
lm(formula = Weight ~ Height + Age, data = cl)

Coefficients:
(Intercept)       Height          Age
   -141.224        3.597        1.278
```

只显示了调用的公式和参数估计结果。

6.13.3 提取信息的通用函数

lm()函数的返回值叫做模型拟合结果对象, 本质上是一个具有类属性值lm的列表, 有model、coefficients、residuals等成员。lm()的结果显示十分简单, 为了获得更多的拟合信息, 可以使用对lm类对象有特殊操作的通用函数, 这些函数包括：

add1 coef effects kappa predict residuals
alias deviance family labels print summary
anova drop1 formula plot proj

下面列出了lm类(拟合模型类)常用的通用函数的简单说明。

- anova(对象1,对象2)　把一个子模型与原模型比较,生成方差分析表。
- coefficients(对象)　返回回归系数(矩阵)。可简写为coef(对象)。

- deviance(对象)　返回残差平方和,如有权重则加权。
- formula(对象)　返回模型公式。
- plot(对象)　绘制模型诊断的几种图, 如残差对预测值图。
- predict(对象,newdata=数据框), predict.gam (对象,newdata=数据框)　有了模型拟合结果后对新数据进行预报。指定的新数据必须与建模时用的数据具有相同的变量结构。函数结果为对数据框中每一观测的因变量预报结果(为向量或矩阵)。

  predict.gam()与predict()作用相同但适用性更广, 可应用于lm、glm和gam的拟合结果。比如,当多项式基函数用了正交多项式时, 加入了新数据导致正交多项式基函数改变, 用predict.gam() 函数可以避免由此引起的偏差。
- print(对象)　简单显示模型拟合结果。一般不用print()而直接键入对象名来显示。
- residuals(对象)　返回模型残差(矩阵),若有权重则适当加权。可简写为resid(对象)。
- summary(对象)　可显示较详细的模型拟合结果。

6.13.4 方差分析

方差方差分析是研究取离散值的因素对一个数

值型指标的影响的经典工具。S进行方差分析的函数是aov(), 格式为aov(公式, data=数据框), 用法与lm()类似, 提取信息的各通用函数仍有效。

我们以前面用过的不同牌子木板磨损比较的数据为例。假设veneer数据框保存了该数据：

```
> veneer
   Brand Wear
1   ACME  2.3
2   ACME  2.1
3   ACME  2.4
4   ACME  2.5
5  CHAMP  2.2
6  CHAMP  2.3
7  CHAMP  2.4
8  CHAMP  2.6
9   AJAX  2.2
10  AJAX  2.0
11  AJAX  1.9
12  AJAX  2.1
13 TUFFY  2.4
14 TUFFY  2.7
15 TUFFY  2.6
16 TUFFY  2.7
17  XTRA  2.3
18  XTRA  2.5
19  XTRA  2.3
20  XTRA  2.4
```

首先我们把每个牌子的木板的磨损情况画盒形图并且放在同一页面中, 作图如下：

```
plot(Wear ~ Brand, data=veneer)
```

见图6.20。这种图可以直观地比较一个变量在多个组的分布, 或者比较几个类似的变量。从图中可以看

出, AJAX牌子较好, TUFFY较差, 其它三个牌子差别不明显。

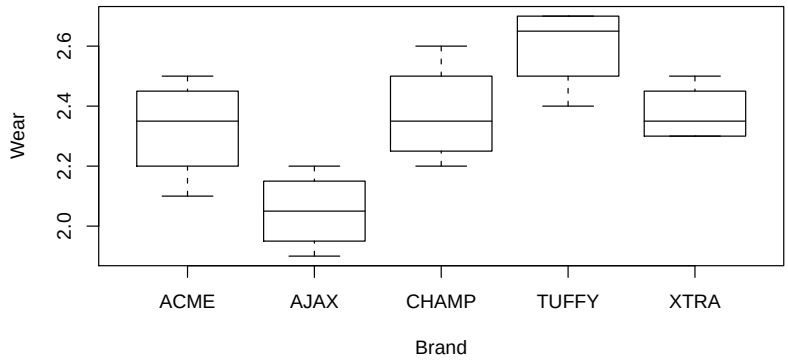


图 6.20: 不同品牌木板磨损量比较

为了检验牌子这个因素对指标磨损量有无显著影响, 只要用aov()函数:

```
> aov.veneer <- aov(Wear ~ Brand, data=veneer)
> summary(aov.veneer)
            Df  Sum Sq Mean Sq F value   Pr(>F)
Brand        4 0.61700 0.15425   7.404 0.001683 **
Residuals   15 0.31250 0.02083
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

可见因素是显著的。

6.14 统计分析实例

下面我们以6.1.2中的那个学生班的情况为例进行一些分析。我们希望了解体重、身高、年龄、性别等变量的基本情况及互相之间的关系。

6.14.1 数据输入

假设数据放在了文本文件c:\work\class.txt中, 没有列标题, 各变量上下对齐。我们先把数据读入一个S数据框对象中：

```
> cl <- read.table("c:/work/class.txt",
+ col.names=c("Name", "Sex", "Age", "Height", "Weight"),
+ row.names="Name")
> cl
        Sex Age Height Weight
Alice     F  13   56.5   84.0
Becka     F  13   65.3   98.0
Gail      F  14   64.3   90.0
 • • • • • • •
William   M  15   66.5  112.0
```

6.14.2 探索性数据分析(EDA)

首先我们先研究各变量的分布情况, 看分布是否接近正态, 有无明显的异常值, 有没有明显的序列相关, 等等。

先用6.12.1中定义的eda.shape()函数来来研究各数值型变量的分布情况。在调用前先把数据框cl连接入当前的搜索路径中以直接使用cl中的变量名：

```
> attach(cl)
> summary(cl)
> eda.shape(Age)
> eda.shape(Height)
> eda.shape(Weight)
> tab.sex <- table(Sex)
> barplot(tab.sex)
```

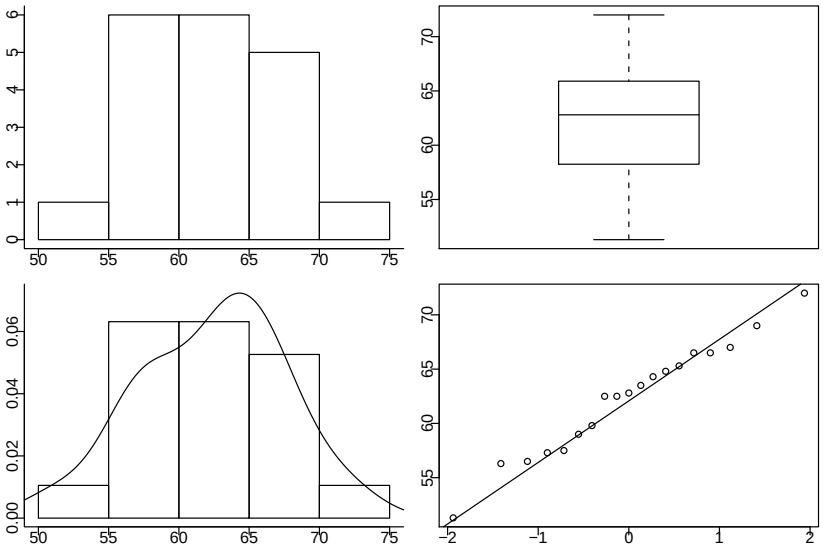


图 6.21: 身高分布探索性数据分析

因为数据是不同个体的观测所以不可能有序列相关, 未画时间序列图。这里给出了身高的分布图(图6.21)。可以看出, 身高和体重都相当接近正态且无明显的异常点, 体重因为取离散值所以直方图不接近正态, 但从核密度估计曲线看仍可作为正态处理。

为了研究数值型变量Weight、Height、Age间的关系, 我们画它们的散点图矩阵:

```
> pairs(cbind(Height, Weight, Age))
```

从散点图矩阵(图6.22)可以看出三个变量之间都可能有线性相关关系。

为了研究因子Sex对其它变量的影响, 可以画Sex不同水平上各变量的盒形图, 如:

```
> oldpar <- par(mfcol=c(1,3))
> boxplot(Weight ~ Sex, ylab="Weight")
```

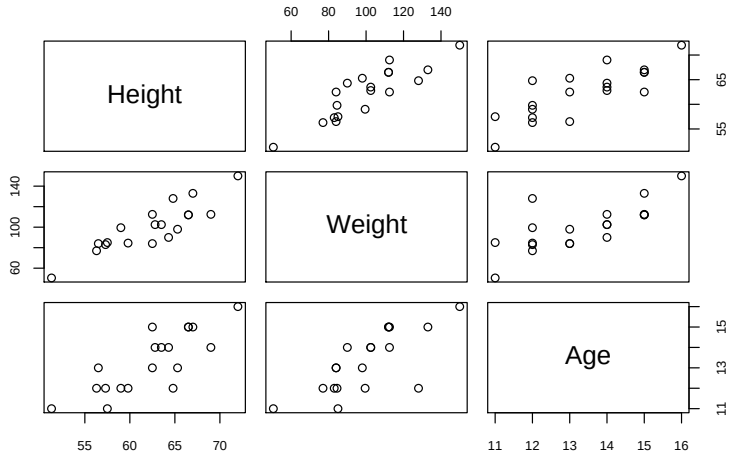


图 6.22: 身高、体重、年龄的散点图矩阵

```
> boxplot(Height ~ Sex, ylab="Height")
> boxplot(Age ~ Sex, ylab="Age")
> par(oldpar)
```

从图6.23可以看出, 男女的体重、身高有明显的差别, 而年龄则差别不明显。我们也可以分不同性别对某一变量分别作图或计算, 这里只要使用象Weight[Sex==”F”], Weight[Sex==”M”]这样的取子集的办法就可以把观测分组。更进一步还可以用函数tapply直接按一个因子对观测分组然后作用某个函数:

```
> tapply(Weight, Sex, hist)
```

为了研究因子Sex的不同水平对其它变量间的相关关系的影响, 可以作协同图:

```
> coplot(Weight ~ Height | Sex)
```

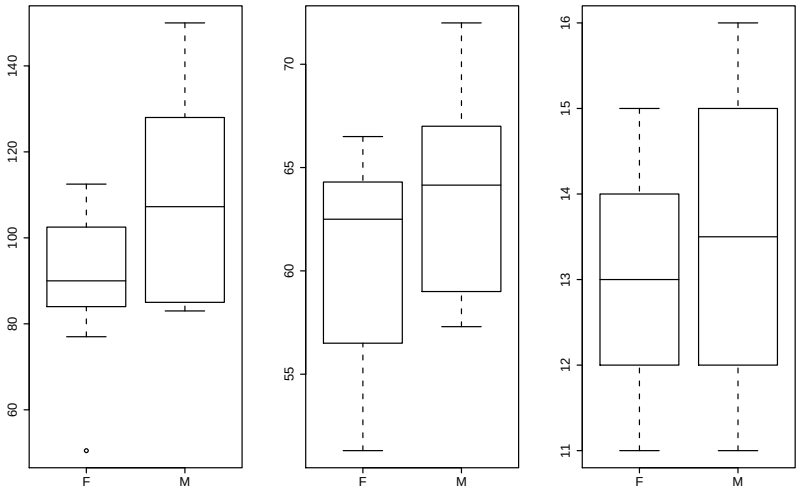


图 6.23: 性别对其他变量的影响

结果图6.12没有反映明显的差别。

6.14.3 组间比较

我们来分析男女的身高有无显著差异, 这是两组比较的问题。上面EDA部分的并排盒形图已经提示男女身高有明显差异, 这里我们用统计假设检验给出统计结论。

男女两组可以认为是独立的, 而且每组内的观测也可以认为是相互独立的。根据EDA结果可以认为两组都来自正态总体。这样, 我们可以使用两样本t检验。因为方差是否相等为止, 我们干脆用不要求方差相等的近似两样本t检验：

```
> t.test(Height ~ Sex)
```

结果p值为0.1652, 按我们一般采用的0.05水平是不显

著的。所以从这组样本看男女的身高没有发现显著差异。

类似可以进行男女体重的比较, p值为0.06799, 也不显著。

6.14.4 回归分析

下面我们研究对体重的预报。从散点图矩阵看, 体重与身高之间有明显的线性相关, 所以我们先拟合一个体重对身高的一元线性回归模型：

```
> lm.fit1 <- lm(Weight ~ Height, data=cl)
> lm.fit1

Call:
lm(formula = Weight ~ Height, data = cl)

Coefficients:
(Intercept)       Height
   -143.027        3.899

> summary(lm.fit1)

Call:
lm(formula = Weight ~ Height, data = cl)

Residuals:
     Min       1Q   Median       3Q      Max
-17.6807  -6.0642   0.5115   9.2846  18.3698

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -143.0269    32.2746  -4.432 0.000366 ***
Height         3.8990     0.5161   7.555 7.89e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 11.23 on 17 degrees of freedom
Multiple R-Squared: 0.7705,     Adjusted R-squared: 0.757
F-statistic: 57.08 on 1 and 17 DF,  p-value: 7.887e-007
```

拟合的模型方程为Weight = -143.0269 + 3.8990×

Height, 复相关系数平方为0.7705, 检验模型的斜率为0的p值为7.887e−007, 可见模型是显著的。对于一元回归, 我们可以在因变量对自变量的散点图上叠加回归直线来看回归拟合的效果：

```
> plot(Weight ~ Height)
> abline(lm.fit1)
```

一般地, lm拟合结果对象的plot()函数可以作出若干张检查拟合效果的图形, R可以作四个图：残差对拟合值图、残差的正态概率图、标准化残差绝对值平方根对拟合值图、Cook距离图。

```
> oldpar <- par(mfrow=c(2,2),
+   mar=c(2.5,2,1.5,0.2), mgp=c(1.2, 0.2, 0))
> plot(lm.fit1)
> par(oldpar)
```

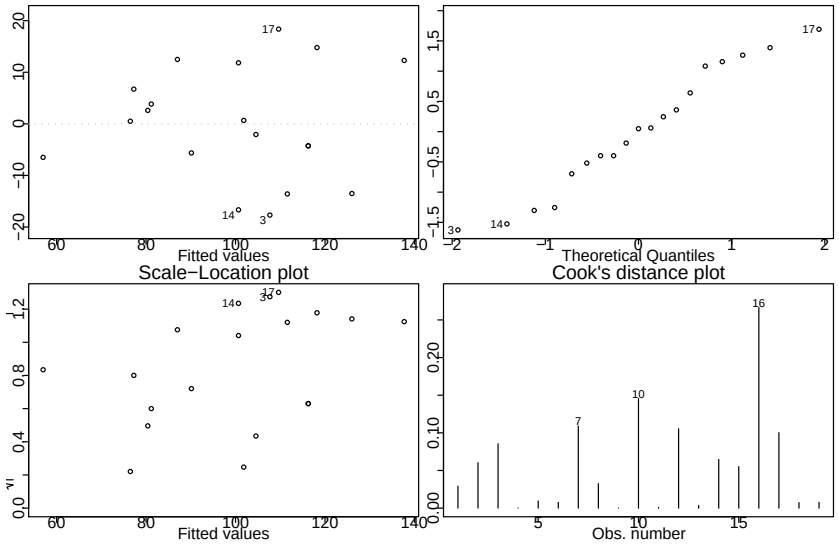


图 6.24: 回归诊断图

见图6.24。如果想每个图都用全窗口来看则不要设置图形参数。四个图中, 残差对拟合值图可以反映残差中残留的结构, 如果模型充分的话残差应该是随机变化没有任何模式的。残差的正态概率图可以检验线性回归假设检验的重要假定—误差项服从正态分布是否合理, 可以看出残差的分布重尾、轻尾、左偏、右偏等情况。标准化残差绝对值平方根对拟合值图可以发现残差的异常值点, 即拟合最差的点。Cook距离衡量每一观测对拟合结果的影响大小, 数值大的为强影响点。图中自动标出了最突出的点。

从lm.fit1的回归诊断图看残差没有明显的模式, 但残差分布有轻尾倾向。没有明显的异常值点。

下面我们看加入其它变量能否进一步改善模型的预报能力。用add1函数可以判断加入新的变量可以改善模型:

```
> add1(lm.fit1, ~ . + Age + Sex)
Single term additions

Model:
Weight ~ Height
       Df Sum of Sq     RSS     AIC
<none>              2142.49   93.78
Age     1     22.39 2120.10   95.58
Sex     1    184.71 1957.77   94.07
```

add1的结果显示一个方差分析表, 各行中<none>一行为不加变量的情况, Age一行为加入一个变量Age后的情况, Sex一行为加入一个变量Sex的情况。各列中DF为此变量的自由度, Sum of Sq为该变量对应的平方和, RSS为加入该变量后的残差平方

和, AIC为加入该变量后的AIC统计量值。AIC较小的模型为较好的, 所以如果加入某个变量后的AIC减小就可以加入此变量。这里加入Age和加入Sex都使AIC变大, 所以不应加入这两个变量。

如果一开始就加入了所有变量, 可以用drop1()函数考察去掉一个变量后AIC是否可以变小：

```
> lm.fit2 <- lm(Weight ~ Height + Age + Sex, data=cl)
> summary(lm.fit2)

Call:
lm(formula = Weight ~ Height + Age + Sex, data = cl)

Residuals:
     Min       1Q   Median       3Q      Max
-19.6540  -6.5737   0.4602   7.6708  20.8515

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -125.1151    33.9038  -3.690  0.00218 **
Height         2.8729     0.9971   2.881  0.01142 *
Age            3.1131     3.2362   0.962  0.35132
SexM           8.7443     5.8350   1.499  0.15472
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 11.09 on 15 degrees of freedom
Multiple R-Squared: 0.8025,     Adjusted R-squared: 0.763
F-statistic: 20.31 on 3 and 15 DF,  p-value: 1.536e-005

> drop1(lm.fit2)
Single term deletions

Model:
Weight ~ Height + Age + Sex
       Df Sum of Sq     RSS     AIC
<none>              1844.01   94.93
Height  1   1020.61 2864.62  101.30
Age     1    113.76 1957.77   94.07
Sex     1    276.09 2120.10   95.58
```

从summary()的结果看Age和Sex就不显著。用drop1()发现去掉Age可以使AIC从94.93变小

为94.07, 所以应该去掉Age。对去掉Age后的模型再用drop1()发现Sex也应该去掉。所以我们最后得到的模型是lm.fit1。

在修改模型或数据改变后重新拟合时还可以使用update函数, 比如要从lm.fit2中去掉Age就可以用：

```
> lm.fit3 <- update(lm.fit2, . ~ . - Age)
```

在自变量个数较多时S提供了一个step()函数用来进行逐步回归, 它从一个初始模型开始自动判断增加或去掉变量, 最后得到较好的模型：

```
> lm.fit0 <- lm(Weight ~ 1, data=cl)
> lm.step <- step(lm.fit0, ~ . + Height + Age + Sex)
Start:  AIC= 119.75
 Weight ~ 1

         Df Sum of Sq    RSS    AIC
+ Height  1    7193.2 2142.5   93.8
+ Age     1    5124.5 4211.2  106.6
+ Sex     1    1681.1 7654.6  118.0
<none>                9335.7  119.7

Step:  AIC= 93.78
 Weight ~ Height

         Df Sum of Sq    RSS    AIC
<none>                2142.5   93.8
+ Sex     1     184.7 1957.8   94.1
+ Age     1      22.4 2120.1   95.6
- Height  1    7193.2 9335.7  119.7
```

得到适当的模型以后, 我们可以用模型进行拟合或预报。拟合只要对模型拟合结果用predict()函数, 如:

```
> predict(lm.fit1)
```

对新数据如果想作预报, 只要在predict函数调用时加入新数据的数据框作为参数：

```
> new.data <- data.frame(Height=c(50, 51.2, 68, 69.7))
> predict(lm.fit1, new.data)
       1        2        3        4
 51.92459  56.60343 122.10714 128.73549
```

6.15 用S作随机模拟计算

作为统计工作者, 我们除了可以用S迅速实现新的统计方法, 还可以用S进行随机模拟。随机模拟可以验证我们的算法、比较不同算法的的优缺点、发现改进统计方法的方向, 是进行统计研究的最有力的计算工具之一。

随机模拟最基本的需要是产生伪随机数, S中已提供了大多数常用分布的伪随机数函数, 可以返回一个伪随机数序列向量。这些伪随机数函数以字母r开头, 比如rnorm()是正态伪随机数函数, runif()是均匀分布伪随机数函数, 其第一个自变量是伪随机数序列长度n。关于这些函数可以参见6.16以及系统帮助文件。下例产生1000个标准正态伪随机数：

```
> y <- rnorm(1000)
```

这些伪随机数函数也可以指定与分布有关的参数, 比如下例产生1000个均值为150、标准差为100的正态伪随机数：

```
> y <- rnorm(1000, mean=150, sd=100)
```

产生伪随机数序列是不重复的, 实际上, S在产生伪随机数时从一个种子出发, 不断迭代更新种子, 所以产生若干随机数后内部的随机数种子就已经改变了。有时我们需要模拟结果是可重复的, 这只要我们保存当前的随机数种子, 然后在每次产生伪随机数序列之前把随机数种子置为保存值即可：

```
> the.seed <- .Random.seed
• • • • • •
> .Random.seed <- the.seed
> y <- rnorm(1000)
```

作为例子, 我们来产生服从一个简单的线性回归的数据。

```
# 简单线性回归的模拟
lm.simu <- function(n){
    # 先生成自变量。假设自变量x的取值范围
    # 在150到180之间,大致服从正态分布。
    x <- rnorm(n, mean=165, sd=7.5)
    # 再生成模型误差。假设服从N(0, 1.2)分布
    eps <- rnorm(n, 0, 1.2)
    # 用模型生成因变量
    y <- 0.8 * x + eps
    return(data.frame(y,x))
}
```

S没有提供多元随机变量的模拟程序, 这里给出一个进行三元正态随机变量模拟的例子。假设要产生三元正态随机向量$X \sim N(\mu, \Sigma)$的n个独立观测, 可以先产生n个服从三元标准正态分布的观测, 放在一个n行3列的矩阵中：

```
U <- matrix(rnorm(3*n), ncol=3, byrow=T)
```

可以认为矩阵U的每一行是一个标准的三元正态分布的观测。设矩阵Σ的Choleski分解为$\Sigma = A'A$, A为上三角矩阵, 若随机向量$\xi \sim N(0, I)$, 则$\mu + A'\xi \sim N(\mu, \Sigma)$。因此, $\mu + A'U'$作为一个三行n列的矩阵每一列都是服从$N(\mu, \Sigma)$分布的, 且各列之间独立。经过转置, 产生的X

```
X <- matrix(rep(mu,n), ncol=3, byrow=T) + U %*% A
```

是一个n行三列的矩阵。

有时模拟需要的计算量很大, 多的时候甚至要计算几天的时间。对于这种问题我们要善于把问题拆分成可以单独计算的小问题, 然后单独计算每个小问题, 把结果保存在S对象中或文本文件中, 最后综合保存的结果得到最终结果。

如果某一个问题需要的计算时间比较长, 我们在编程时可以采用以下的技巧：每隔一定时间就显示一下任务的进度, 以免计算已经出错或进入死循环还不知道；应该把中间结果每隔一段时间就记录到一个文本文件中(cat()函数可以带一个file参数和append参数, 对这种记录方法提供了支持), 如果需要中断程序, 中间结果可能是有用的, 有些情况下还可以根据记录的中间结果从程序中断的地方继续执行。

因为S是一个解释型语言, 所以有些需要大量迭代的任务纯用S编程效率很低。为此我们可以把主要部分用S编程, 把需要迭代的部分用C、FORTRAN或C++编程, 在S中调用由其他语言编译得到的DLL库。这里只简单介绍如何在MS Windows 版本的R中调用C编译得到的DLL库。

注意Borland公司免费提供了一个MS Windows下的C++编译程序, 有一个自由软件的DevC++集成编程环境也可以进行C和C++编辑和编译。

假设我们写了一个计算卷积的C程序:

```
void convolve(double *a, int *na,
      double *b, int *nb, double *ab){
    int i, j, nab = *na + *nb - 1;
    for(i = 0; i < nab; i++)
    ab[i] = 0.0;
    for(i = 0; i < *na; i++)
    for(j = 0; j < *nb; j++)
    ab[i + j] += a[i] * b[j];
}
```

注意C函数的所有自变量都是指针, 这是因为S的基本数据类型是一维数组。假设此C程序编译成了一个DLL库testdll.dll放在当前工作目录。注意编译时要生成Windows的控制台DLL而不是窗口环境的DLL。在资源管理器中查看可以发现DLL中输出了函数'convolve'。为了在R中调用这个C函数, 需要如下步骤:

```
conv <- function(a, b){
    dll.filename <- file.path(paste("testdll", .Platform$dynlib.ext, se
    if(!is.loaded(symbol.C("convolve"), PACKAGE="testdll")){
        dyn.load(dll.filename)
    }
    .C("convolve", as.double(a), as.integer(length(a)),
       as.double(b), as.integer(length(b)),
       ab=double(length(a)+length(b)-1))$ab
}
```

这个S函数把C函数"convolve"进行了包装。其中, 函数is.loaded(symbol.C(*函数名*))检查DLL中的C函

数是否已经调入了R中, 如果没有调入就要用dyn.load(*DLL文件名*)调入。因为在Windows环境下和Unix环境下DLL库的扩展名不同所以对文件名做了兼容性处理。为了调用C函数, 使用".C"函数。".C"函数的第一个参数是DLL中的C函数名(注意可能带下划线), 其他参数是C函数的自变量。所有从R传给C的自变量都要用as.double和as.integer包裹起来。".C"函数返回一个列表,列表元素为传入的自变量经过函数运行后的值, 如果调用".C"时给某些自变量加了名字(如上面的ab)则返回的列表中将使用这些名字。

对于反复调用的函数可以设法预先调入动态链接库而不是每次在函数内判断。

6.16 S常用函数参考

这一节分类列出常用的函数, 需要时可以参看帮助。

6.16.1 基本

一、数据管理

vector:向量 numeric:数值型向量 logical:逻辑型向量 character:字符型向量 list:列表 data.frame:数据框 c:连接为向量或列表 length:求长度 subset:求子集 seq, from:to, sequence:等差序列 rep:重复 NA:缺失值 NULL:空对象 sort, order, unique, rev:排序 unlist:展平列表 attr, attributes:对象属性

mode, typeof:对象存储模式与类型 names:对象的名字属性

二、字符串处理

character:字符型向量 nchar:字符数 substr:取子串 format, formatC:把对象用格式转换为字符串 paste, strsplit:连接或拆分 charmatch, pmatch:字符串匹配 grep, sub, gsub:模式匹配与替换

三、复数

complex, Re, Im, Mod, Arg, Conj:复数函数

四、因子

factor:因子 codes:因子的编码 levels:因子的各水平的名字 nlevels:因子的水平个数 cut:把数值型对象分区间转换为因子 table:交叉频数表 split:按因子分组 aggregate:计算各数据子集的概括统计量 tapply:对“不规则”数组应用函数

6.16.2 数学

一、计算

+, -, *, /, , %%, %/%:四则运算 ceiling, floor, round, signif, trunc, zapsmall:舍入 max, min, pmax, pmin:最大最小值 range:最大值和最小值 sum, prod:向量元素和, 积 cumsum, cumprod, cummax, cummin:累加、累乘 sort:排序 approx和approx fun:插值 diff:差分 sign:符号函数

二、数学函数

abs, sqrt:绝对值, 平方根 log, exp, log10, log2:对数与指数函数 sin, cos, tan, asin, acos, atan, atan2:三角函数 sinh, cosh, tanh, asinh, acosh, atanh:双曲函数 beta, lbeta, gamma, lgamma, digamma, trigamma, tetragamma, pentagamma, choose, lchoose:与贝塔函数、伽玛函数、组合数有关的特殊函数 fft, mvfft, convolve:富利叶变换及卷积 polyroot:多项式求根 poly:正交多项式 spline, splinefun:样条差值 besselI, besselK, besselJ, besselY, gamma-Cody:Bessel函数 deriv:简单表达式的符号微分或算法微分

三、数组

array:建立数组matrix:生成矩阵 data.matrix:把数据框转换为数值型矩阵 lower.tri:矩阵的下三角部分 mat.or.vec:生成矩阵或向量 t:矩阵转置 cbind:把列合并为矩阵 rbind:把行合并为矩阵 diag:矩阵对角元素向量或生成对角矩阵 aperm:数组转置 nrow, ncol:计算数组的行数和列数 dim:对象的维向量 dimnames:对象的维名 row/colnames:行名或列名 %*%:矩阵乘法 crossprod:矩阵交叉乘积(内积) outer:数组外积 kronecker:数组的Kronecker积 apply:对数组的某些维应用函数 tapply:对“不规则”数组应用函数 sweep:计算数组的概括统计量 aggregate:计算数据子集的概括统计量 scale:矩阵标

准化 matplot:对矩阵各列绘图 cor:相关阵或协差阵 Contrast:对照矩阵 row:矩阵的行下标集 col:求列下标集

四、线性代数

solve:解线性方程组或求逆 eigen:矩阵的特征值分解 svd:矩阵的奇异值分解 backsolve:解上三角或下三角方程组 chol:Choleski分解 qr:矩阵的QR分解 chol2inv:由Choleski分解求逆

五、逻辑运算

<, >, <=, >=, ==, !=:比较运算符 !, &, &&, |, ||, xor():逻辑运算符 logical:生成逻辑向量 all, any:逻辑向量都为真或存在真 ifelse():二者择一 match, %in%:查找 unique:找出互不相同的元素 which:找到真值下标集合 duplicated:找到重复元素

六、优化及求根

optimize, uniroot, polyroot:一维优化与求根

6.16.3 程序设计

一、控制结构

if, else, ifelse, switch:分支 for, while, repeat, break, next:循环 apply, lapply, sapply, tapply, sweep:替代循环的函数。

二、函数

function:函数定义 source:调用文件 call:函数调用 .C, .Fortran:调用C或者Fortran子程序的动态链接库。 Recall:递归调用 browser, debug, trace, traceback:程序调试 options:指定系统参数 missing:判断虚参是否有对应实参 nargs:参数个数 stop:终止函数执行 on.exit:指定退出时执行 eval, expression:表达式计算 system.time:表达式计算计时 invisible:使变量不显示 menu:选择菜单(字符列表菜单)

其它与函数有关的还有:delay, delete.response, deparse, do.call, dput, environment, , formals, format.info, interactive, is.finite, is.function, is.language, is.recursive, match.arg, match.call, match.fun, model.extract, name, parse, substitute, sys.parent, warning, machine。

三、输入输出

cat, print:显示对象 sink:输出转向到指定文件 dump, save, dput, write, write.table:输出对象 scan, read.table, load, dget:读入

四、工作环境

ls, objects:显示对象列表rm, remove:删除对象 q, quit:退出系统 .First, .Last:初始运行函数与退出运行函数。 options:系统选项 ?, help, help.start, apropos:帮助功能 data:列出数据集

6.16.4 统计计算

一、统计分布

每一种分布有四个函数:d—density(密度函数), p—分布函数, q—分位数函数, r—随机数函数。比如, 正态分布的这四个函数为dnorm, pnorm, qnorm, rnorm。下面我们列出各分布后缀, 前面加前缀d、p、q或r就构成函数名:

norm:正态, t:t分布, f:F分布, chisq:卡方(包括非中心), unif:均匀, exp:指数, weibull:威布尔, gamma:伽玛, beta:贝塔, lnorm:对数正态, logis:逻辑分布, cauchy:柯西, binom:二项分布, geom:几何分布, hyper:超几何, nbinom:负二项, pois:泊松, signrank:符号秩, wilcox:秩和, tukey:学生化极差

二、简单统计量

sum, mean, var, sd, min, max, range, median, IQR(四分位间距)等为统计量, sort, order, rank与排序有关, 其它还有ave, fivenum, mad, quantile, stem等。

三、统计检验

R中在ctest包中已实现的有shapiro.test, t.test, wilcox.test, chisq.test, prop.test, 等等。

四、多元分析

cor, cov.wt, var:协方差阵及相关阵计算 biplot, biplot.princomp:多元数据biplot图 cancor:典

则相关　　princomp:主成分分析　　hclust:谱系聚类
kmeans:k-均值聚类　　cmdscale:经典多维标度

其它有dist, mahalanobis, cov.rob。

五、时间序列

ts:时间序列对象　　diff:计算差分　　time:时间序列的采样时间　　window:时间窗

在ts包中实现了acf等函数。

六、统计模型

lm, glm, aov:线性模型、广义线性模型、方差分析

练习

1. (a) 写出元素为3, -1.5, 3E-10的向量。
   (b) 写出从3开始每次增加3,长度为100的向量。
   (c) 写出(0, 2)重复10次的向量。
   (d) 对向量x,写出其元素大于等于0小于1的条件。
   (e) 对向量x,写出其元素都等于0的条件。
   (f) 写出包含12个月份名称的向量。
   (g) 生成一个包含文件名tab1.txt到tab18.txt的字符串向量。
   (h) 写出包含方程$z^6 = 1$的根的向量,并写出其幅角的余弦和正弦值。

2. 设x为一个长100的整数向量。比如, x <− floor(10*runif(100))。
   (a) 显示x第21到30号元素。
   (b) 把x第31,35,39号元素赋值为0。
   (c) 显示x中除了第1号和第50号的元素之外的子集。
   (d) 列出x中个位数等于3的元素。
   (e) 列出x中个位数等于3的元素的下标位置。
   (f) 给x的每一个元素加上名字, 为x1到x100。
   (g) 求x的平均值并求每一个元素减去平均值后的离差, 计算x的离差平方和及元素的平方和。
   (h) 把x从大到小排序。计算x的10%分位数到90%分位数之间的距离。
3. 定义一个维数为(3, 4, 2)的数组其第一层(第三下标为1)取从1开始的奇数, 第二层取从2开始的偶数。显示每一层的第2行元素。把第(1,1,1),(2,2,2), (2, 2, 1)号元素赋值为零。把第一层加上100,把第二层加上200。分别计算第一层和第二层的平均值。
4. 对线性模型$Y = X\beta + \varepsilon$,写出当X满秩时计算β的S表达式。写出估计ε的方差的S表达式。
5. 把SASUSER.GPA数据中的SEX, SATM, SATV分别输入到S中。计算不同性别的人数, 并计算每一组的平均SATM分。把这些变量组合成一个列表。把SASUSER.GPA数据输入为S的数据框。

6. 把语句x <− floor(runif(100))所生成的向量保存到一个文本文件中, 数据项用空格和换行分隔。从此文件中读入数据到向量y中。

7. 设x是一个长度为n的向量,写一段程序,计算x的长度为s的滑动和：

$$S_x(t) = \sum_{i=0}^{s-1} x_{t-i}, \quad t = s, s+1, \ldots, n$$

8. 写一个AR(1)的模拟函数：

$$x_t = a + bx_{t-1} + \varepsilon_t, t = 1, 2, \ldots, n, \mathrm{Var}(\varepsilon_t) = \sigma^2$$

函数的参数为n、a、b、x_0和σ, 缺省时n=100, a=0, b=1, x_0=0, σ=1。