苏州大学数学科学学院

统计计算与 SAS 软件包实验报告

姓名：__熊雄__学号：__1907402030__年级：__19 级__日期：__2021/11/10__

实验 6　ODS、宏及 SQL

实验目的：掌握 ODS 和 SAS 宏

实验内容：

1 PROC PRINTTO 过程的作用及其例子

2 举例说明 ODS LISTING 的作用

3 举例说明 ODS TRACE 的作用

4 举例说明 ODS HTML 的作用

5 举例说明 ODS OUTPUT 的作用

6 SAS 宏变量及宏的定义，调用

7 多个&号的作用

8 熟悉并例举 SQL 中各种语句。

9 用 SQL 完成实验三的 exam12.xls 中的工作

结果与分析

实 验 过 程

一、 PROC PRINTTO 过程的作用及其例子

1.1 PROC PRINTTO 过程的作用

Reset the SAS log and procedure output destinations to default. PROC PRINTTO routes subsequent logs and procedure output to their default destinations and closes both of the current files.

1.2 举例

以下代码代表将日志输出到 D 盘下的 homework6.txt，new 代表若有同名文件就替换掉，若无 new，则表示将日志内容添加到文件中。

```
proc printto log="D:\homework6.txt" new;
run;
```

二、 举例说明 ODS LISTING 的作用

2.1 ODS LISTING 的作用

ODS 中，数据就像游客，通过各种过程步而来，ODS 处理每一个数据集并发送到目的地。实际上，不同的 ODS 类型就是目的地，当达到目的地时，而数据的样式是由模板决定。如果没有指定目的地，那么你的数据默认发往“列表 listing”。

LISTING 目标的显示方式：

✓ 字符文本在 OUTPUT 窗口显示；

✓ 图形在 GRAPH1 窗口显示。

2.2 举例

在 SAS 中输入以下代码：

```
filename out ' D:\listing_1.txt';
ods listing file=out; /*将结果输出到listing_1.txt文件中*/
proc freq data=sashelp.class;
run;
ods listing;
```

提交后可以在 D 盘找到 listing_1.txt，里面的内容与结果查看器里的一致，如下图所示：

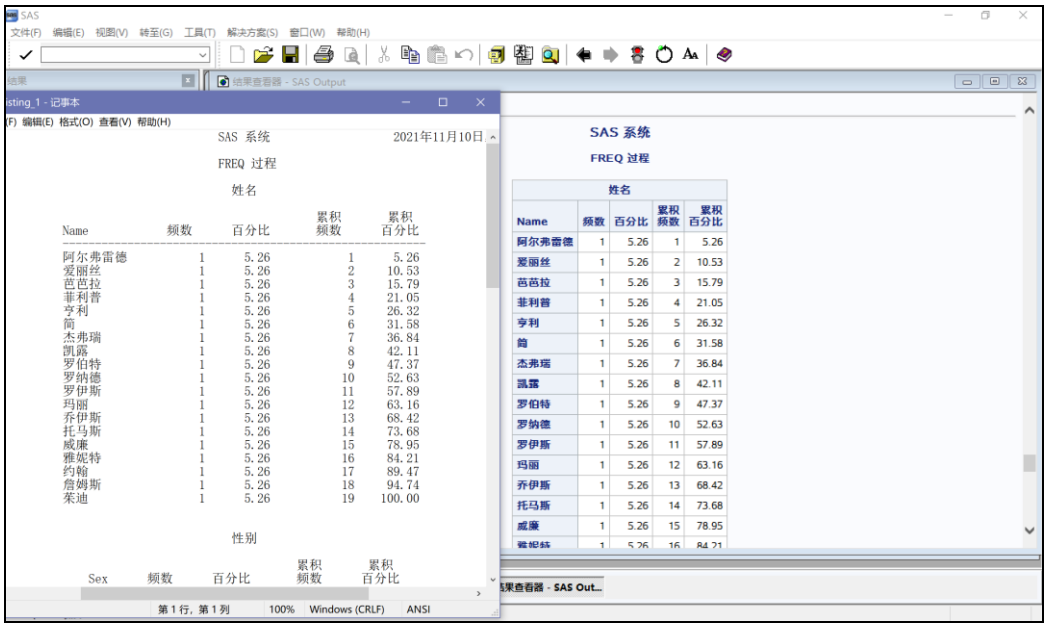


Figure 1：ODS LISTING 代码运行结果示意图

三、 举例说明 ODS TRACE 的作用

3.1 ODS TRACE 的作用

为了对输出对象进行选择或剔除，首先必须查询 PROC 步的输出中都包含了哪些对象。要查询某个 PROC 步的输出对象，就得使用以下的 ODS TRACE 语句：

```
ODS TRACE ON </选项>;
ODS TRACE OFF;
```

3.2 举例

打开追踪功能运行 CONTENTS 过程：

```
ods trace on/ label;
proc contents data = sashelp.class;
run;
ods trace off;
```

日志如下：

```
93   ods listing;
94   ods trace on/ label;
95   proc contents data = sashelp.class;
96   run;


Output Added:
-------------
名称:        Attributes
标签:        属性
模板:        Base.Contents.Attributes
路径:        Contents.DataSet.Attributes
标签路径:    'Contents PROCEDURE'.'SASHELP.CLASS'.'属性'
-------------

Output Added:
-------------
名称:        EngineHost
标签:        引擎/主机信息
模板:        Base.Contents.EngineHost
路径:        Contents.DataSet.EngineHost
标签路径:    'Contents PROCEDURE'.'SASHELP.CLASS'.'引擎/主机信息'
-------------

Output Added:
-------------
名称:        Variables
标签:        变量
模板:        Base.Contents.Variables
路径:        Contents.DataSet.Variables
标签路径:    'Contents PROCEDURE'.'SASHELP.CLASS'.'变量'
-------------
NOTE: "PROCEDURE CONTENTS"所用时间（总处理时间）：
      实际时间           0.04 秒
      CPU 时间           0.03 秒


97   ods trace off;
```

Figure 2：CONTENTS 过程运行的日志

四、 举例说明 ODS HTML 的作用

4.1 ODS HTML 的作用

通常情况下，如果用户没有主动通过代码或者系统设置关闭传送目标 HTML，HTML 会一直处于打开状态。使用 HTML 语句可以自己定义 HTML 输出文件的框架、内容、主体，其使用的语法如下：

```
ODS HTML BODY='BODY文件名.HTML';
  SAS 代码;
ODS HTML CLOSE;
```

其中，选项 BODY=指明 HTML 输出中包含主体的文件名称。这里也可以使用选项 FILE=来代替 BODY=。

4.2 举例

```
ods html body = 'out.html';
proc univariate data = sashelp.class;
  var age;
run;
ods html close;
```

五、 举例说明 ODS OUTPUT 的作用

5.1 ODS OUTPUT 的作用

将输出窗口 OUTPUT 的输出对象转换成 SAS 数据集，其格式为：

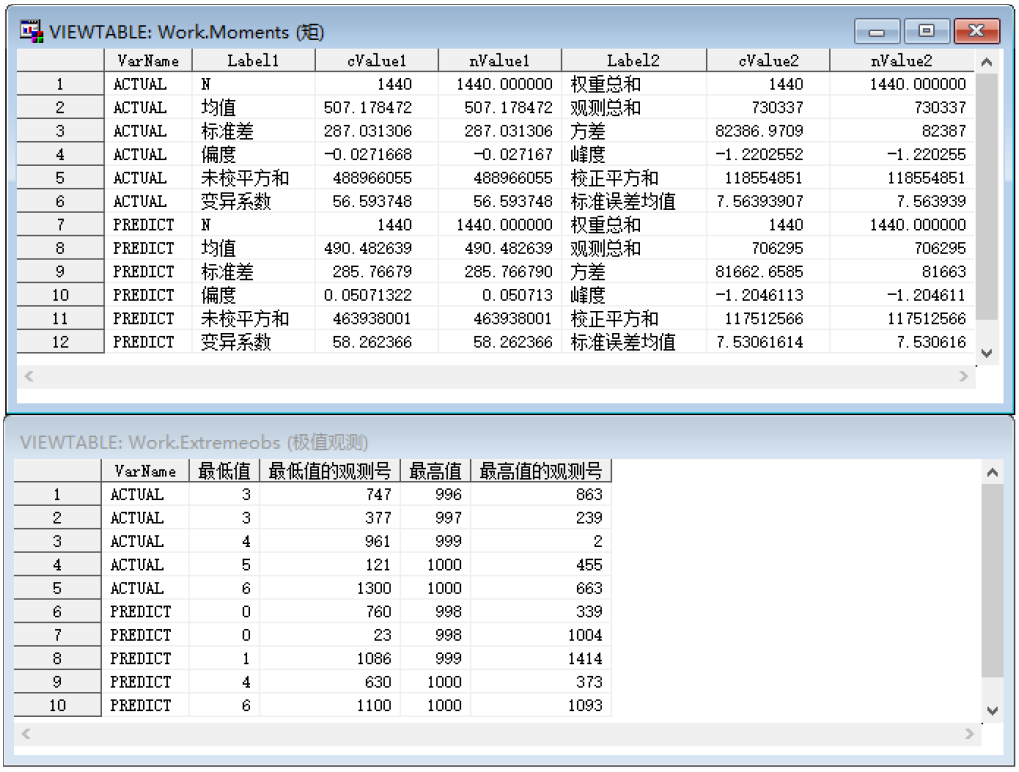
```
ODS OUTPUT 输出对象 1=数据集 1 输出对象 2=数据集 2…;
```

5.2 举例

输入以下代码：

```
ods output ExtremeObs = work.ExtremeObs Moments = work.Moments;
proc univariate data = sashelp.prdsale;
   var actual predict;
run;
ods output close;
```

在上面的代码中，只指定了部分路径，如果打开输出对象查询跟踪功能，我们会发现部分路径为 ExtremeObs 的输出对象有两个，分别为 Univariate.ACTUAL.ExtremeObs 和 Univariate.PREDIC.ExtremeObs，但最后只生成了一个数据集 work.ExtremeObs，系统会将具有相同部分路径的不同输出对象迭加起来，输送到同一个数据集中。数据集 work.Monments 也是同样的。数据集的内容如下图所示：



VIEWTABLE: Work.Moments (矩)

	VarName	Label1	cValue1	nValue1	Label2	cValue2	nValue2
1	ACTUAL	N	1440	1440.000000	权重总和	1440	1440.000000
2	ACTUAL	均值	507.178472	507.178472	观测总和	730337	730337
3	ACTUAL	标准差	287.031306	287.031306	方差	82386.9709	82387
4	ACTUAL	偏度	-0.0271668	-0.027167	峰度	-1.2202552	-1.220255
5	ACTUAL	未校平方和	488966055	488966055	校正平方和	118554851	118554851
6	ACTUAL	变异系数	56.593748	56.593748	标准误差均值	7.56393907	7.563939
7	PREDICT	N	1440	1440.000000	权重总和	1440	1440.000000
8	PREDICT	均值	490.482639	490.482639	观测总和	706295	706295
9	PREDICT	标准差	285.76679	285.766790	方差	81662.6585	81663
10	PREDICT	偏度	0.05071322	0.050713	峰度	-1.2046113	-1.204611
11	PREDICT	未校平方和	463938001	463938001	校正平方和	117512566	117512566
12	PREDICT	变异系数	58.262366	58.262366	标准误差均值	7.53061614	7.530616

VIEWTABLE: Work.Extremeobs (极值观测)

	VarName	最低值	最低值的观测号	最高值	最高值的观测号
1	ACTUAL	3	747	996	863
2	ACTUAL	3	377	997	239
3	ACTUAL	4	961	999	2
4	ACTUAL	5	121	1000	455
5	ACTUAL	6	1300	1000	663
6	PREDICT	0	760	998	339
7	PREDICT	0	23	998	1004
8	PREDICT	1	1086	999	1414
9	PREDICT	4	630	1000	373
10	PREDICT	6	1100	1000	1093

Figure 3：数据集内容

六、 SAS 宏变量及宏的定义与调用

7.1 宏变量的定义

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代入宏变量的值。在不混淆的时候可以省略以上格式中的小数点。宏变量代换完全是一种字符串的替换，学过 C 语言后可以发现这与 C 中的宏定义是同一类型。

7.2 宏变量的调用举例

对于以下代码：

```
%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.;
```

另外需要注意的是，单引号里面的宏变量不会被解析，所以如果使用引号并且希望引号中的宏变量被解析，则必须使用双引号。

7.3 宏程序（简称：宏）的定义

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 宏有点像是子程序，但是一定注意其中的子程序参数和宏变量都只有字符串替换的意义。简单的宏的定义方法如下：

```
%MACRO 宏名称;
  宏文本;
%MEND 宏名称;
```

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;
```

7.4 宏的调用

调用宏时，宏名称后面不需要加分号。此外，宏可以在 SAS 程序中除数据行以外的任何地方被调用。例如，假设一个程序中多次要用到当前的系统时间，则可以考虑写一个调用时间的宏，代码如下：

```
%macro time;
  %put The current time is %sysfunc(time(),timeampm.);
%mend time;
```

```
其他程序代码片段1
%time
其他程序代码片段2
%time
...
```

七、 多个&号的作用

在宏编程中，有时候需要间接地调用宏变量，间接调用宏变量是通过多个“&”符号来实现的。宏处理器对多个“&”的处理规则是：从左到右扫描，若宏变量前仅含有一个“&”，则解析该宏变量；否则将相邻的两个“&”替换成一个“&”，重复上述扫描过程。例如我们有以下的代码：

```
%let ID = 9;
%let NAME9 = JACK;
%put &&&NAME&ID;
```

该代码在%PUT 语句中间接引用了宏变量 NAME9。

八、 熟悉并例举 SQL 中各种语句

8.1 单表操作

表 sashelp.cars 包含了汽车生产商制造的各种型号汽车的数据。下面使用 SQL 语句输出以下 3 列：Make（生产商）、Model（型号）以及 MSRP（建议零售价）。同时生成一个新列 Tax（税金），其值为 MSRP*0.06，输出 MRSP 不大于 4000 的行。

```
proc sql;
title "Generating A New Column";
select cars.Make, cars.Model, cars.MSRP, cars.msrp * 0.06 as tax
from sashelp.cars;
where calculated tax <= 2400;
quit;
```

需要注意的是，代码中使用了 TITLE 语句为输出的报表生成标题。一般来说，TITLE 语句的位置可以在 PROC SQL 之前，也可位于 SELECT 从句之前。且对于新生成的列，用户必须在前面加上关键字 CALCULATED。本段代码部分输出结果如下图：

Generating A New Column

Make	Model	MSRP	tax
Acura	MDX	$36,945	2216.7
Acura	RSX Type S 2dr	$23,820	1429.2
Acura	TSX 4dr	$26,990	1619.4
Acura	TL 4dr	$33,195	1991.7
Acura	3.5 RL 4dr	$43,755	2625.3
Acura	3.5 RL w/Navigation 4dr	$46,100	2766
Acura	NSX coupe 2dr manual S	$89,765	5385.9
Audi	A4 1.8T 4dr	$25,940	1556.4
Audi	A41.8T convertible 2dr	$35,940	2156.4
Audi	A4 3.0 4dr	$31,840	1910.4
Audi	A4 3.0 Quattro 4dr manual	$33,430	2005.8
Audi	A4 3.0 Quattro 4dr auto	$34,480	2068.8
Audi	A6 3.0 4dr	$36,640	2198.4
Audi	A6 3.0 Quattro 4dr	$39,640	2378.4
Audi	A4 3.0 convertible 2dr	$42,490	2549.4
Audi	A4 3.0 Quattro convertible 2dr	$44,240	2654.4

Figure 4：代码的部分输出结果

8.2 多表操作

8.2.1 内部 join

```
data math;
input number MathGrade$;
datalines;
0     90
1     95
2     90
2     90
2     90
3     85
run;
data chinese;
input level ChineseGrade$;
datalines;
1     85
2     80
2     80
3     75
4     90
run;
```

```
proc sql;
select * from chinese, math
where math.number = chinese.level;
quit;
```

运行的结果如下：

Generating A New Column

level	ChineseGrade	number	MathGrade
1	85	1	95
2	80	2	90
2	80	2	90
2	80	2	90
2	80	2	90
2	80	2	90
2	80	2	90
3	75	3	85

Figure 5：内部 join 代码运行结果图

8.2.2 左外部连接

```
proc sql;
select * from math a left join chinese b
on a.number=b.level;
quit;
```

运行结果如下：

Generating A New Column

number	MathGrade	level	ChineseGrade
0	90	.	
1	95	1	85
2	90	2	80
2	90	2	80
2	90	2	80
2	90	2	80
2	90	2	80
2	90	2	80
3	85	3	75

Figure 6：左外部连接代码运行结果

8.2.3 右外部连接

```
proc sql;
select * from math a right join chinese b
```

```
on a.number=b.level;
quit;
```

运行结果如下：

Generating A New Column

number	MathGrade	level	ChineseGrade
1	95	1	85
2	90	2	80
2	90	2	80
2	90	2	80
2	90	2	80
2	90	2	80
2	90	2	80
3	85	3	75
.		4	90

Figure 7：右外部连接代码运行结果图

8.2.4 完全外部连接

```
proc sql;
select * from math a full join chinese b
on a.number=b.level;
quit;
```

运行结果如下：

Generating A New Column

number	MathGrade	level	ChineseGrade
0	90	.	
1	95	1	85
2	90	2	80
2	90	2	80
2	90	2	80
2	90	2	80
2	90	2	80
2	90	2	80
3	85	3	75
.		4	90

Figure 8：完全外部连接代码运行结果图

九、 用 SQL 完成实验三的 exam12.xls 中的工作

9.1 将 data1.txt 转化为 SAS 数据集

输入以下代码：

```
filename data1 'E:\sas class\实验三\data1.txt';
data ex1;
   infile data1 firstobs=2 dlm ='09'x ; /*从第二行开始读起，分隔符为Tab*/
   length name $15.;
   input
  name$
  sex$
  age;
run;
proc print data = ex1;
run;
```

提交后输出如下：

SAS 系统

Obs	name	sex	age
1	阿尔弗雷德	男	14
2	爱丽丝	女	13
3	芭芭拉	女	13
4	菲利普	男	16
5	亨利	男	14
6	简	女	12
7	杰弗瑞	男	13

Figure 9：**数据集** work.ex1

9.2 将 data2.csv 转化为 SAS 数据集

输入以下代码：

```
filename data2 'E:\sas class\实验三\data2.csv';
data ex2;
   infile data2 firstobs=2 dlm =','; /*从第二行开始读起，分隔符为,*/
   length name $15.;
   input
  name$
  height
  weight;
run;
proc print data = ex2;
run;
```

提交后输出如下：

SAS 系统

Obs	name	height	weight
1	阿尔弗雷德	69.0	112.5
2	爱丽丝	56.5	84.0
3	芭芭拉	65.3	98.0
4	菲利普	72.0	150.0
5	亨利	63.5	102.5
6	简	59.8	84.5
7	杰弗瑞	62.5	84.0

Figure 10：数据集 work.ex2

9.3 利用 sql 过程合并 SAS 数据集 ex1 与 ex2

输入以下代码：

```
proc sql;
select ex1.name,ex1.age, ex1.sex, ex2.height,ex2.weight
from work.ex1 full join work.ex2
on ex1.name = ex2.name;
quit;
```

运行结果如下：

创建数据集

name	age	sex	height	weight
阿尔弗雷德	14	男	69	112.5
爱丽丝	13	女	56.5	84
芭芭拉	13	女	65.3	98
菲利普	16	男	72	150
亨利	14	男	63.5	102.5
简	12	女	59.8	84.5
杰弗瑞	13	男	62.5	84

Figure 11：合并后的数据集