Программирование на языке SAS BASE

Лекция 3:Макросы и SQL

Звежинский Дмитрий,
SAS Russia/CIS
dmitry.zvezhinsky@sas.com

Замечания об ошибках и опечатках просьба направлять лектору.

Макроподстановки

- Мы научились писать код на языке SAS Base, но он не очень «гибкий».
  - У нас нет возможности организовать условное ветвление программы между частями кода
  - … и циклы, внутри которых выполняются шаги программы.
  - Мы не можем повторно использовать отлаженную часть кода.
  - Нет возможности вводить в программу параметры для повторного использования кода в разных частях программы. Если нужно поменять один и тот же кусок кода – только средствами редактора (search - replace)
- Чтобы решить эти задачи, в SAS есть надстройка над языком SAS Base, которая называется SAS Macro.
- Help: **SAS® 9.3 Macro Language Reference**
- По сути это продвинутый (и автоматизированный) вариант «copy»+ «paste». Путь разработчика – написать и отладить программу **без** макросов, а потом **добавить** макросы.

Как работает макропроцессор

- Существуют специальные инструкции, чтобы управлять макроподстановками в код программы, условными переходами и динамическим созданием кода (когда используются подстановки с параметрами)
- Макропроцессор выполняет обработку программы **до** её компиляции (синтаксической проверки) и исполнения. После работы макропроцессора должен получиться код на языке SAS BASE без каких-либо макро-инструкций. Другими словами, синтаксис и логика макро-языка и SAS BASE изолированы друг от друга. Мы должны научиться их «дружить» между собой.
- Первый пример макроинструкции – это создание и подстановка макропеременной.

Пример: распечатаем **n** первых наблюдений из набора данных, где **n** будет задаваться макропеременной.

Решение без макросов: использовать опцию к набору данных «obs=»

```
proc print data=ecprg1.donate(obs=10);
run;
```

Как работает макропроцессор

- Создание макропеременной (одним оператором можно создать только одну переменную):

```
%let numb_of_obs = 10 ;
```

- Название – не более 32 символов английского алфавита (+ цифры и знак подчеркивания), регистр не важен, начинается с буквы или знака подчеркивания.
- Всё, что стоит между знаком «=» и знаком «;» является значением макропеременной numb_of_obs.
- Любое значение макропеременной – это текст, который потом будет куда-то подставлен. Разделения на символьный и числовой тип – нет.
- Математические выражения не выполняются
- Регистр хранимого текста сохраняется
- Минимальная длина 0 символов, максимальная – 65534 симв.
- Пробелы с начала и с конца значения удаляются.
- Значение макропеременной «живёт» до конца сеанса (или завершения процесса SAS)
- …или до применения макрофункции %symdel ;

Пользовательские макропеременные

- Вывод в log значения макропеременной:

```
%put &numb_of_obs ;
```

- Подстановка макропеременной в программу:

```
proc print data=ecprg1.donate(obs=&numb_of_obs);
run;
```

- Log:

```
15          %let numb_of_obs = 10 ;
16          %put &numb_of_obs ;
10
17          proc print data=ecprg1.donate(obs=&numb_of_obs);
18          run;
```

- По умолчанию все средства для отладки выключены, и мы не видим сообщений от макропроцессора.
- Макропеременные можно подставлять одну в другую:

```
15          %let x=aaa;
16          %let x=&x bbb;
17          %put &x;
aaa bbb
```

- Задание явной границы (.) в коде для названия макропеременной:

```
15          %let x=aaa;
16          %let x=&x.bbb;
17          %put &x;
aaabbb
```

Отладка макропеременных

- Вывод в log только пользовательских переменных:

```
%PUT _USER_ ;
```

- Вывод в log всех макропеременных:

```
%PUT _ALL_ ;
```

- Включить глобальную опцию symbolgen:

```
15          options symbolgen;
16          %let x=aaa;
17          %let x=&x.bbb;
SYMBOLGEN:  Macro variable X resolves to aaa
18          %put &x;
SYMBOLGEN:  Macro variable X resolves to aaabbb
aaabbb
```

- Выключить её, если макрос отлажен: `options nosymbolgen;`
- Log всегда содержит информацию об ошибках разрешения макроподстановок:

```
17          %let x=&x11.bbb;
WARNING: Apparent symbolic reference X11 not resolved.
18          %put &x11;
WARNING: Apparent symbolic reference X11 not resolved.
&x11
```

Работа с макропеременными

- Подстановка переменных в программу:

```
15          %let library= ecprg1 ;
16          %let dataset= donate ;
17          data temp1;
18             set &library..&dataset  ;
19          run;

NOTE: There were 16 observations read from the data set ECPRG1.DONATE.
NOTE: The data set WORK.TEMP1 has 16 observations and 3 variables.
```

- Подстановка переменной в символьные константы:

Одинарные кавычки – макропроцессор не подставляет значение макропеременной, двойные кавычки – подставляет:

```
15          %let text1= Privet! ;
16          data  null ;
17            put "&text1";
18            put '&text1';
19          run;
Privet!
&text1
```

- Автоматические макропеременные:

SYSDATE, SYSDATE9, SYSDAY, SYSTIME – дата, день недели и время, когда был запущен процесс SAS (могут отличаться от текущих!)

SYSLAST – название самого последнего набора данных, который был создан

SYSERR – код возврата шага DATA или PROC (0=ошибок нет)

SYSLIBRC – код возврата оператора LIBNAME (создание ссылки на библиотеку, 0=ошибок нет)

Макроопределения

- Макропроцессор может динамически создавать код, используя заданные шаблоны для макроподстановки.
- Шаг 1. Определение макрофункции.

```
%macro my_macro_fun;
  proc print data=ecprg1.donate;
    var employee_id qtr amount;
  run;
%mend;
```

Макропроцессор запомнит этот кусок кода

- Шаг 2. Вызов макрофункции: если макроопределение было занесено в макропроцессор, оно подставляется в сеанс и выполняется.

```
%my_macro_fun
```

Компиляция и выполнение шага происходит, когда будут разрешены (=подставлены) все макропеременные, то есть можно писать что-то типа:

```
%macro my_macro_fun1;
  proc print data=ecprg1.donate;
%mend;
%macro my_macro_fun2;
  var employee_id qtr amount;
%mend;
%my_macro_fun1
%my_macro_fun2
run;
```

Вопрос: чем отличаются макроопределения от макропеременных?
(ответа пока не было)

Порядок выполнения макроопределения

```
15          options mcompilenote=all;
16          %macro test1;
17            data _null_;
18              a=&ttt;
19            run;
20          %mend;
NOTE: The macro TEST1 completed compilation without errors.
      5 instructions 88 bytes.
21          %test1
MLOGIC(TEST1):  Beginning execution.
NOTE: Line generated by the invoked macro "TEST1".
21           data _null_;     a=&ttt;   run;
                              _
                              22
MPRINT(TEST1):   data _null_;
WARNING: Apparent symbolic reference TTT not resolved.
MPRINT(TEST1):   a=&ttt;
MPRINT(TEST1):   run;
```

Этот код компилируется (запоминается) макропроцессором, т.к. он не содержит ошибки макро-синтаксиса, но в нем могут присутствовать другие виды ошибок (разрешение макро переменных, или ошибки SAS BASE)

Обратите внимание, что макро-переменные разрешаются после выполнения макроопределения

Макропеременной «ttt» не существует, ошибка в разрешении этой переменной привела к появлению в программе SAS непонятной конструкции

Отладка макроопределений

- `Options mcompilenote=ALL;`

```
NOTE: The macro MYMACRO completed compilation without errors.
```

- `Options mlogic;`

  Помогает отлаживать условные переходы в макропрограммах

- `Options mprint;`

  Печатает код, который генерируется макросом

- Список макроопределений

```
proc catalog cat=work.sasmac1;
contents;
run;quit;
```

6	MY_MACRO_FUN1	MACRO	17May13:16:02:35	17May13:16:02:35
7	MY_MACRO_FUN2	MACRO	17May13:16:02:35	17May13:16:02:35

- Список макропеременных: служебное представление sashelp.vmacro

VMACRO

Filter and Sort | Query Builder | Data | Describe | Graph | Analyze

	scope	name	offset	value
7	GLOBAL	_CLIENTPROJE...	0	
8	GLOBAL	_SASPROGRA...	0	
9	GLOBAL	TEXT1	0	Privet!
10	GLOBAL	DATASET	0	donate

- Отключите все отладочные опции (`Options mcompilenote=NONE nomlogic nomprint;`) после отладки.

Вызов макроопределений

- Макропеременные можно подставлять в макроопределения.

```
Options mcompilenote=ALL mlogic mprint symbolgen;
%let dataset=ecprg1.donate ;
%let vars=employee_id qtr amount;
%macro my_macro_fun1;
  proc print data=&dataset;
    var &vars;
  run;
%mend;
%my_macro_fun1
```

Определяем макропеременные

Определяем макроопределение

Вызов макроопределения

LOG:

```
16          %let dataset=ecprg1.donate ;
17          %let vars=employee_id qtr amount;
18          %macro my_macro_fun1;
19            proc print data=&dataset;
20              var &vars;
21            run;
22          %mend;
NOTE: The macro MY_MACRO_FUN1 completed compilation without errors.
      5 instructions 104 bytes.
23          %my_macro_fun1
MLOGIC(MY_MACRO_FUN1):  Beginning execution.
SYMBOLGEN:  Macro variable DATASET resolves to ecprg1.donate
MPRINT(MY_MACRO_FUN1):   proc print data=ecprg1.donate;
SYMBOLGEN:  Macro variable VARS resolves to employee_id qtr amount
MPRINT(MY_MACRO_FUN1):   var employee_id qtr amount;
MPRINT(MY_MACRO_FUN1):   run;

NOTE: . . .
MLOGIC(MY_MACRO_FUN1):  Ending execution.
```

Параметры в макроопределениях

- В макроопределения можно передавать параметры.

```
%macro my_macro_fun1(dataset, vars);
  proc print data=&dataset;
    var &vars;
  run;
%mend;
```

1 аргумент 2 аргумент

```
%my_macro_fun1(ecprg1.donate,employee_id qtr amount)
```

LOG:

```
NOTE: The macro MY_MACRO_FUN1 completed compilation without errors.
      9 instructions 232 bytes.
20         %my_macro_fun1(ecprg1.donate,employee_id qtr amount)
MLOGIC(MY_MACRO_FUN1):  Beginning execution.
MLOGIC(MY_MACRO_FUN1):  Parameter DATASET has value ecprg1.donate
MLOGIC(MY_MACRO_FUN1):  Parameter VARS has value employee_id qtr amount
SYMBOLGEN:  Macro variable DATASET resolves to ecprg1.donate
MPRINT(MY_MACRO_FUN1):   proc print data=ecprg1.donate;
SYMBOLGEN:  Macro variable VARS resolves to employee_id qtr amount
MPRINT(MY_MACRO_FUN1):   var employee_id qtr amount;
MPRINT(MY_MACRO_FUN1):   run;

NOTE: ...

MLOGIC(MY_MACRO_FUN1):  Ending execution.
```

У макропеременных есть области видимости, см help -> Examples of Macro Variable Scopes. Областью видимости переменной можно принудительно управлять с помощью %global (перем. видна во всем макрокоде) и %local (только в данном макросе)

Параметры в макроопределениях

- В макроопределениях можно задать параметры с помощью двух нотаций (позиционная и по ключевому слову). значение параметра по умолчанию, а также передавать параметры в макроопределение в произвольном порядке

```
%macro my_macro_fun1(dataset, var1=employee id,var2=qtr, var3= );
  proc print data=&dataset;
    var &var1 &var2 &var3 ;
  run;
%mend;
%my_macro_fun1(ecprg1.donate)
```

LOG:

```
. . .
MLOGIC(MY_MACRO_FUN1): Parameter DATASET has value ecprg1.donate
MLOGIC(MY_MACRO_FUN1): Parameter VAR1 has value employee_id
MLOGIC(MY_MACRO_FUN1): Parameter VAR2 has value qtr
MLOGIC(MY_MACRO_FUN1): Parameter VAR3 has value
```

Obs	Employee_ID	Qtr
1	11036	2
2	11036	3

```
%my_macro_fun1(ecprg1.donate,var2=amount)
```

LOG:

```
. . .
MLOGIC(MY_MACRO_FUN1): Parameter VAR2 has value amount
MLOGIC(MY_MACRO_FUN1): Parameter VAR1 has value employee_id
MLOGIC(MY_MACRO_FUN1): Parameter VAR3 has value
```

Obs	Employee_ID	Amount
1	11036	20
2	11036	25

Условный переход

```
%if &var = CHANGE %then
      %do;
...ПОДСТАНОВКА1
      %end;
%else %if &var = SAME %then
      %do;
...ПОДСТАНОВКА2
      %end;
```

- В условии, которое проверяет оператор %if, должны находиться макропеременные и/или макрофункции, то есть то, с чем может «разобраться» макропроцессор на момент вызова этого выражения.
- Оператор %if можно вызвать только внутри макроопределения:

```
15          %let var=1;
16          %if &var =0 %then
ERROR: The %IF statement is not valid in open code.
17          %do;
```

- Проверяемое выражение должно возвращать «число», в противном случае макропроцессор выдаёт ошибку.

Операции, которые можно использовать для %if

- Список (IN) (не работает без `options minoperator;`)

```
15          %macro test;
16          %let var=A;
17          %if &var IN A B C D %then %put true;
18          %else %put false;
19          %mend;
20          %test
true
```

- 0 = false, <>0 = true

```
15          %macro test;
16          %if 0 %then %put true;
17          %else %put false;
18          %mend;
19          %test
False
```

- Условие вычисляется после разрешения макропеременных

```
16          %macro test;
17          %let var=1;
18          %let oper= ~;
19          %let var2=0;
20          %if &var &oper &var2 %then %put true;
21          %else %put false;
22          %mend;
23          %test
true
```

SAS(R) Macro Language: Reference->Macro Expressions->Defining Arithmetic and Logical Expressions

Макрофункции

- %SYSEVALF(expression<, conversion-type>)

Вычисление арифметических и логических операций (в т.ч. с нецелыми числами)

```
15          %put %sysevalf(4+0.9);
4.9
```

- %SYSFUNC (function(argument-1 <...argument-n>)<, format>)

Выполняет функцию SAS и возвращает результат

```
16          %put %sysfunc(ceil(0.9));
1
17          %put %sysfunc(today());
19498
```

- %NRSTR (character-string)

Экранировка служебных символов, в том числе «&», «,»

```
15          %let xstring1=&VARVAR;
16          %let xstring2=%nrstr(&VARVAR);
17          %put &xstring1;
WARNING: Apparent symbolic reference VARVAR not resolved.
&VARVAR
18          %put &xstring2;
&VARVAR
```

- %SUBSTR (argument, position<, length>)

```
19          %put %SUBSTR (argument, 4, 4 ) ;
umen
```

* см. SAS® 9.3 Macro Language Reference

Работа с макропеременными

- Если макропеременная содержит цифру, что с ней можно сделать?
- Арифметические и логические операции:

```
%let a=8;
%let b=&a+8; /* wrong!*/
%put &b;
%let b=%sysevalf(&a+8); /* 16 */
%put &b;
%let c=%sysevalf(&a>&b); /* 0 */
%let d=%sysevalf(&a<&b); /* 1 */
%put &c &d ;
```

- Посчитать какую-нибудь функцию из SAS BASE:

```
%let pi=3.1415926535;
%let a=%sin(&pi); /* what is "%sin"? */
%let a=%sysfunc(sin(&pi));
%put &a; /* almost 0 */
%let a=%sysfunc(tan(&pi/2));
%put &a; /* something goes wrong */
```

```
WARNING: An argument to the function TAN referenced by the %SYSFUNC or %QSYSFUNC macro
function is out of range.
NOTE: Mathematical operations could not be performed during %SYSFUNC function execution
```

Циклы

- Можно использовать только внутри макроопределений.
- Справка: **%DO, Iterative Statement ; %DO %UNTIL Statement ; %DO %WHILE Statement**
- Пример: создание 10 наборов данных, внутри каждого – по 10 наблюдений

```
%macro test;
  %do i=1 %to 10;
    data test&i;
      do i=1 to 10;
      output;
      end;
     run;
  %end;
%mend;
%test
```

- В примере шаг DATA с 10 разными названиями набора данных будет выполнен 10 раз.
- В циклах SAS MACRO можно использовать макроопределения и макропеременные.

Внешнее хранилище макросов

- По окончанию сеанса работы с SAS все макропеременные и макропрограммы удаляются из своих хранилищ. Можно ли их не пересоздавать, если они будут нужны в другом сеансе?
- Можно, причем есть несколько механизмов. Например, для макроса с предыдущего слайда:

```
options mstored sasmstore=ecprg1;
%macro test / store;
    ...
```

- Если у вас подключена библиотека ecprg1, вы можете увидеть в ней появившийся каталог sasmacr, где хранится макрос.

  Или воспользуйтесь proc catalog:

```
proc catalog c=ecprg1.sasmacr;
 contents;
 run;quit;
```

Contents of Catalog ECPRG1.SASMACR

#	Name	Type	Create Date	Modified Date	Des
1	TEST	MACRO	10/02/2014 06:44:41	10/02/2014 06:44:41	

- Чтобы пользоваться «скомпилированными» макросами, вам достаточно установить опции mstored и sasmstore (с перечислением библиотек с каталогом sasmacr или каких-то ваших каталогов, где нужно искать макросы)

Symputx, symget

- Вы можете создать макропеременную на шаге DATA

```
15          data _null_;
16            set ecprg1.donate end=_tab;
17            if _tab then call symputx('tabnobs',_N_);
18          run;
NOTE: There were 16 observations read from the data set ECPRG1.DONATE.
...
19         %put &tabnobs;
16
```

- Вы можете прочитать макропеременную на шаге DATA

```
data donatelast;
  lastobs=input(symget('tabnobs'),5.);
  put lastobs;
  set ecprg1.donate point=lastobs;
  output;
  stop;
run;
```

Подумайте, чем отличается эта программа:

```
data donatelast;
  set ecprg1.donate point=&tabnobs;
  output;
  stop;
run;
```

- Symget возвращает значение символьного типа, даже если в макропеременной хранилось число (в этом случае нужно выполнить преобразование типов).
- Ключ “point=“ оператора “set” позволяет прочесть наблюдение с указанным номером во входящем наборе данных, при этом указывается название переменной числового типа (где хранится номер наблюдения)

%include

- Вы можете “подставлять” в сеанс файлы с кодом SAS, и выполнять этот код. Условие – чтобы подставляемые файлы виделись на сервере, где установлен SAS.

```
%include 'c:/workshop/myprogram.sas';
```

- Чтобы отлаживать этот оператор на разных уровнях вложенности, есть специальная опция:

```
%INCLUDE 'c:/workshop/myprogram.sas' /SOURCE2;
```

Если эта опция включена, в log появляется код, который подставляется из указанного файла и выполняется.

Пользовательские процедуры (FCMP)

- Вы можете создавать собственные функции и подпрограммы (call routines). Их можно использовать в разных шагах SAS (DATA, PROC SQL, в процедурах SAS/OR (задачи оптимизации), SAS/STAT).
- Особенность этого подхода - наличие централизованного хранилища процедур и функций (хотя у макросов тоже есть возможность сохранить скомпилированный код в каталог и пользоваться им коллективно). Причем можно разделять «области видимости» функций.
- Использует синтаксис, похожий на шаг DATA (но! со своими особенностями)

Пример

```
proc fcmp outlib=work.fun1.fun1;

  subroutine plus1(ind);
    outargs ind;
         ind=ind+1;
  endsub;

  function myfactor(x);
    if (x=1) then return (1);
    else return (x*myfactor(x-1));
  endsub;

quit;

options cmplib=work.fun1;

data test;
  n=14;
  call plus1(n);
  f15=myfactor(n);
  f15a=fact(n);
run;
```

Work.fun1 – это название набора данных, куда будут записаны (в специальной форме) функция и подпрограмма. Третья часть этой опции – название package'a.
Подпрограмма plus1 (один входной аргумент числового типа,
Тот же аргумент будет возвращен из подпрограммы – явно указываем с помощью outargs)
endsub; после каждой функции/подпрограммы

Напишем свою функцию, вычисляющую факториал.
Функция принимает и возвращает число.
Return () – инструкция, которая возвращает результат и выходит из функции.

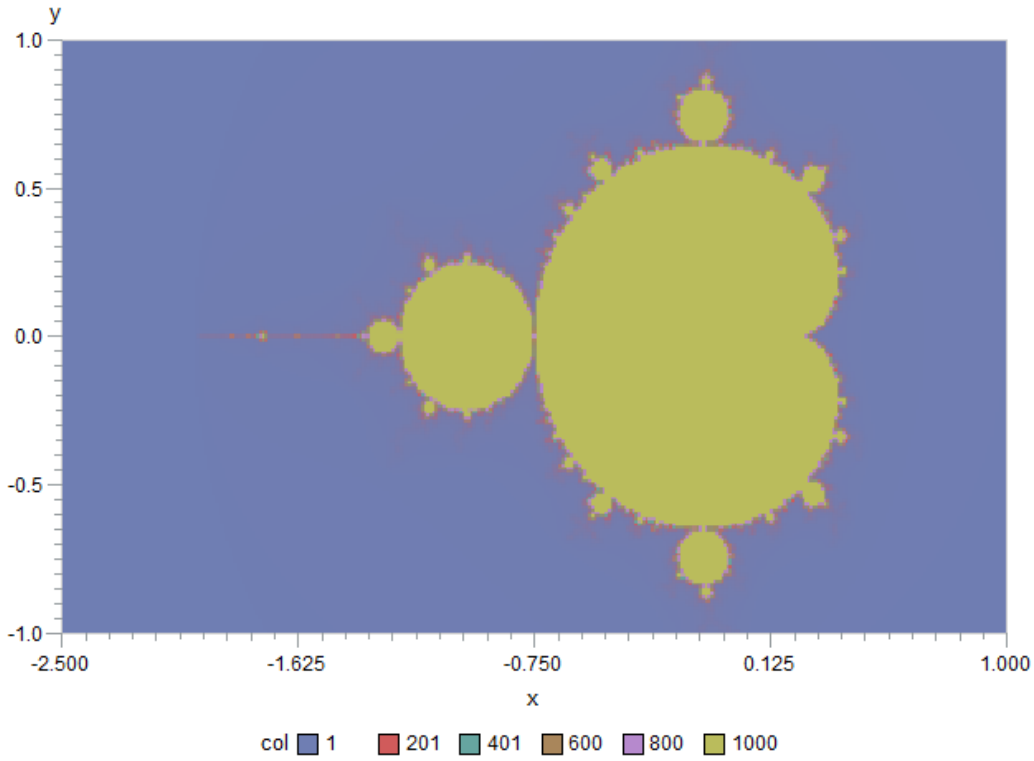
Опция cmplib – где (и в каком порядке) будет производиться поиск пользовательских функций

После вызова call, n=15

Fact = это встроенная функция.

Пример 2 (Mandelbrot set plot)*

```
proc fcmp outlib=work.funcs.temp ;
function mnd (x0,y0); *returns color;
/*For each pixel on the screen do:
  x0 = scaled x coordinate of pixel (must be scaled to lie somewhere in the mandelbrot X scale (-2.5, 1)
  y0 = scaled y coordinate of pixel (must be scaled to lie somewhere in the mandelbrot Y scale (-1, 1)*/
  x = 0;
  y = 0;
  iteration = 0;
  max_iteration = 1000;
  do while ( (x*x + y*y < 2*2)  AND  (iteration < max_iteration) );
    xtemp = x*x - y*y + x0;
    y = 2*x*y + y0;
    x = xtemp;
    iteration = iteration + 1;
  end;
  return(iteration);
endsub;
run;
options cmplib=work.funcs.temp;
data plotted;
  do x=-2.5 to 1 by 0.01;
    do y=-1 to 1 by 0.01;
              col=mnd(x,y);
              output;
            end;
  end;
run;
PROC GCONTOUR DATA = plotted;
            PLOT y * x = col /
            PATTERN;
run;
quit;
```



* From wikipedia. График не строится в SAS U

Просмотр своих функций и их отладка

- Просмотр кода функции (если он не зашифрован)

```
options cmplib=work.fun1;
proc fcmp
outlib=work.fun1.fun1;
listfunc myfactor;
listfunc plus1;
run;
```

- Отладка функции – пользуемся log из функции.

```
 function myfactor(x);
    put x=;
    if (x=1) then return (1);
    else return (x*myfactor(x-1));
endsub;
```

- Отладка функции – ещё один способ (не работает в SAS U, в Enterprise Guide см. окно Results-Listing, виден стек вызовов)

```
proc fcmp outlib=work.fun1.fun1 TRACE ;
  function myfactor(x);
    if (x=1) then return (1);
    else return (x*myfactor(x-1));
  endsub;
a=myfactor(15);
put a=;
run;
```

} Можно вызывать функции прямо из proc fcmp

Процедура SQL

- В SAS можно управлять структурой наборов данных с помощью языка запроса SQL.
- SQL и шаг DATA дополняют друг друга. Подумайте, чем можно проще и эффективнее решать вашу задачу.
- Я не буду рассказывать про сам SQL (в ваших вузах вы можете найти людей, которые сделают это гораздо лучше. Я его когда-то учил с помощью http://sql-ex.ru/).
- Справка: **SAS® 9.3 SQL Procedure User's Guide**
- Совместимость: PROC SQL follows most of the guidelines set by the American National Standards Institute (ANSI) in its implementation of SQL. However, it is not fully compliant with the current ANSI standard for SQL. The SQL research project at SAS has focused primarily on the expressive power of SQL as a query language. Consequently, some of the database features of SQL have not yet been implemented in PROC SQL.
- В общем, это ещё один способ работать с наборами данных SAS.
- Синтаксис:

```
proc sql <опции>;
        <Выражения языка SQL;>
quit;
```

Создание таблицы и её вывод в отчет

- Select выводит результаты SQL-запроса в отчет:

```
proc sql;
  select * from ecsql1.qtr1_2007;
quit;
```

- Create table создаёт набор данных:

```
proc sql;
  create table table1 as
  select t1.order_id, t1.customer_id, t1.order_type
                              from ecsql1.qtr1_2007 as t1
  where t1.order_type=2;
quit;
```

Избранные опции к proc sql:

```
proc sql outobs=10;
WARNING: Statement terminated early due to OUTOBS=10 option.
proc sql inobs=10;
19           select * from ecsql1.qtr1_2007;
WARNING: Only 10 records were read from ECSQL1.QTR1_2007 due to INOBS= option.
proc sql noexec;
NOTE: Statement not executed due to NOEXEC option.
proc sql _method _tree;
```

(информация от планировщика, официально не документировано, см. Paper CS-11 The SQL Optimizer Project: _Method and _Tree in SAS®9.1, Russ Lavery)

Объединение таблиц

- «По горизонтали»

```
proc sql;
  select * from ecsql1.qtr1_2007 as t1, ecsql1.customer as t2
  where t1.customer_id = t2.customer_id;
quit;
proc sql;
  select * from ecsql1.qtr1_2007 as t1 join ecsql1.customer as t2
  on t1.customer_id = t2.customer_id;
quit;
```

- «По вертикали», причем в новой таблице будут все переменные из входящих таблиц (независимо от того, есть ли они сразу в обеих таблицах)

```
proc sql;
  create table qtr12 as
  select * from ecsql1.qtr1_2007
  outer union corr
  select * from ecsql1.qtr2_2007
  ;
quit;
```

См. Combining Queries with Set Operators

Использование функций SAS

- В SQL-запросах можно использовать встроенные и пользовательские функции.

```
proc fcmp outlib=work.fun1.fun1;
  function myfun(id) $ 4;
    str_id=put(id,10.);
       return(substr(str_id,1,4));
  endsub;
run;
options cmplib=work.fun1.fun1;
proc sql;
  select order_id, myfun(order_id) as first_4_lett from
       ecsql1.qtr1_2007 ;
quit;
```

Сведение данных

- Подсчет агрегатов

```
proc sql;
  select order_type, avg(quantity) format=3.1 label="Average!",
     sum(total_retail_price) format=dollar10. label="Sum"
  from ecsql1.order_fact
  group by 1; *possible in group by, order by;
quit;
```

- Having: фильтрация по агрегатам:

```
proc sql;
  select order_type, count(distinct customer_id), sum(total_retail_price)
  from ecsql1.order_fact
  group by order_type
  having sum(total_retail_price) > 30000;
quit;
```

- Calculated: Операции с агрегатами

```
proc sql;
  select order_type, count(distinct customer_id) as people,
                     sum(total_retail_price) as summ,
                     calculated summ/calculated people as ratio
  from ecsql1.order_fact
  group by order_type;
quit;
```

Сведение данных. Remerge.

- Эта особенность SAS SQL помогает сводить исходные данные и агрегаты без вложенных запросов.
- Задача: найти людей, которые имеют максимальную зарплату (salary) в своей группе (gender).
- Обычный SQL: с помощью вложенных запросов

```
proc sql;
  select gender,salary from ecsql1.sales as t1,
  (select gender, max(salary) as maxsalary
  from ecsql1.sales
  group by gender) as t2
  where t1.gender=t2.gender and t1.salary = t2.maxsalary;
quit;
```

- SAS SQL:

```
proc sql;
  select gender, salary,
              max(salary) as ms
  from ecsql1.sales
  group by gender
  having ms=salary;
quit;
```

```
proc sql;
  select gender, salary
  from ecsql1.sales
  group by gender
  having salary = max(salary);
quit;
```

```
NOTE: The query requires remerging summary statistics back with the original
data.
```

Использование макропеременных

- С точки зрения макропроцессора, Proc SQL – обычная процедура, поэтому подстановка макропеременных происходит без проблем:

```
%let stat=min;
proc sql;
  select * from ecsql1.sales
      group by gender
      having salary=&stat.(salary);
quit;
```

- В proc sql вы можете создавать макропеременные:

```
proc sql;
  select avg(salary) into :macro_sal
      from ecsql1.sales;
quit;
%put macro_sal=&macro_sal;
```

- И несколько макропеременных (много трюков, см. справку по INTO Clause):

```
proc sql;
select avg(salary),gender into :macro_sal1-:macro_sal2,:macro_gen1-:macro_gen2
      from ecsql1.sales
      group by gender;
quit;
%put macro_sal=&macro_sal1 macro_gen=&macro_gen1;
%put macro_sal=&macro_sal2 macro_gen=&macro_gen2;
```

Использование макроопределений

```
options mprint mlogic symbolgen;
%MACRO QTR12;
proc sql;
  %do i=1 %to 2;
  select * from ecsql1.qtr&i._2007;
  %end;
quit;
%MEND;
%QTR12

MLOGIC(QTR12):  Beginning execution.
MPRINT(QTR12):   proc sql;
MLOGIC(QTR12):  %DO loop beginning; index variable I; start value is 1; stop value is 2; by
value is 1.
SYMBOLGEN:  Macro variable I resolves to 1
MPRINT(QTR12):   select * from ecsql1.qtr1_2007;
MLOGIC(QTR12):  %DO loop index variable I is now 2; loop will iterate again.
SYMBOLGEN:  Macro variable I resolves to 2
MPRINT(QTR12):   select * from ecsql1.qtr2_2007;
MLOGIC(QTR12):  %DO loop index variable I is now 3; loop will not iterate again.
MPRINT(QTR12):   quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.05 seconds
. . .
MLOGIC(QTR12):  Ending execution.
```

Задания

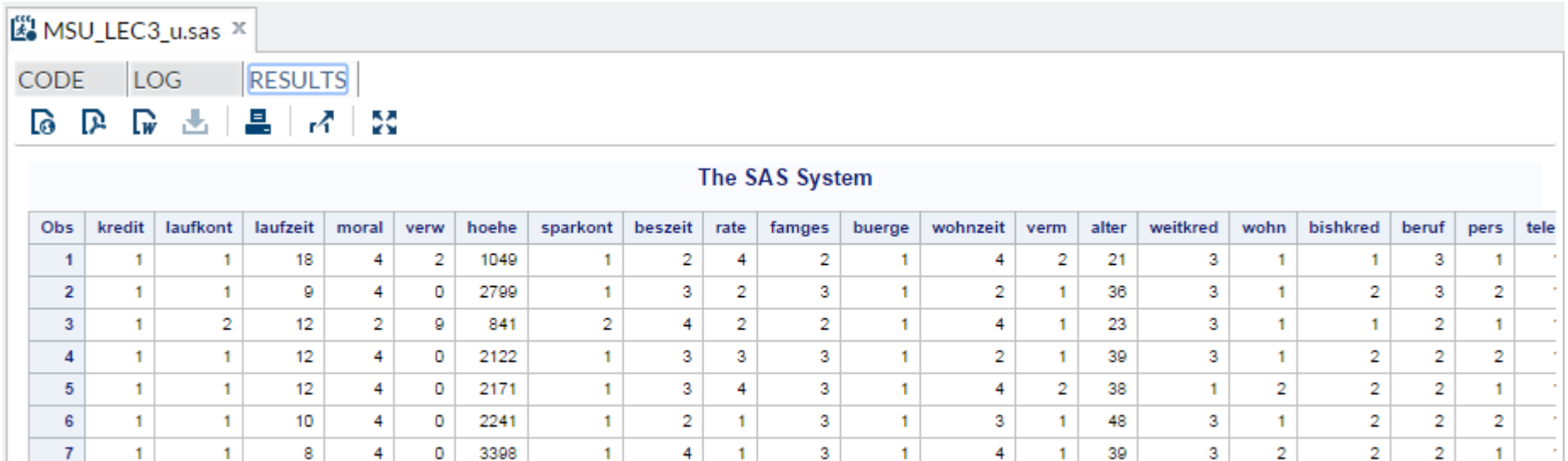
1) Откуда можно взять набор данных? Давайте загрузим (средствами SAS) что-нибудь из интернета и импортируем это в набор данных!

Посмотрите подготовленную программу в конце файла MSU_LEC3_u.sas. Проверьте, работает ли часть кода до PROC IMPORT. (может не работать – тогда требуется указать опции вашего подключения к интернету, например, прокси с помощью опции proxy='http://www.xxx.com‘ или что-то ещё).

Допишите PROC IMPORT и распечатайте новый набор данных.

Добавьте опцию, которая автоматически подхватывает названия из первой строчки этого файла.

Подсмотрите snippet для импорта CSV-файла и справку.


MSU_LEC3_u.sas

CODE | LOG | RESULTS

The SAS System

Obs	kredit	laufkont	laufzeit	moral	verw	hoehe	sparkont	beszeit	rate	famges	buerge	wohnzeit	verm	alter	weitkred	wohn	bishkred	beruf	pers	tele
1	1	1	18	4	2	1049	1	2	4	2	1	4	2	21	3	1	1	3	1	
2	1	1	9	4	0	2799	1	3	2	3	1	2	1	36	3	1	2	3	2	
3	1	2	12	2	9	841	2	4	2	2	1	4	1	23	3	1	1	2	1	
4	1	1	12	4	0	2122	1	3	3	3	1	2	1	39	3	1	2	2	2	
5	1	1	12	4	0	2171	1	3	4	3	1	4	2	38	1	2	2	2	1	
6	1	1	10	4	0	2241	1	2	1	3	1	3	1	48	3	1	2	2	2	
7	1	1	8	4	0	3398	1	4	1	3	1	4	1	39	3	2	2	2	1	

Задания

2) В библиотеке ECPRG1 есть набор данных LOOKUP_COUNTRY. Создать функцию, которая из столбцов LABEL и START сформирует строку типа “Andorra – AD“ (т.е. просто вставляет между ними дефис). Примените эту функцию в SQL-запросе, который печатает всю таблицу в отчет.

Задания

3) Пусть у нас есть таблица с произвольным числом числовых столбцов (с однотипными названиями).

* Посмотрите подготовленную программу в конце файла MSU_LEC3_u.sas. Она создаёт таблицу wide_table.

* Напишите программу (последовательность шагов data/proc), которая сама узнаёт, сколько в этой таблице столбцов с названием “col*”, и создаёт новый набор данных, содержащий сумму этих столбцов.

* Ещё один столбец содержит сумму только тех столбцов, у которых имя заканчивается на четную цифру.

Подсказка: воспользуйтесь служебным представлением sashelp.vcolumn

```
proc print data=sashelp.vcolumn;
 where libname="WORK";
run;
```

Задания

4) Есть вот такой макрос:

```
%macro test(variables,list);
 proc sql;
 select &variables from ecprg1.employee_addresses
 where city in ( "&list" );
 quit;
%mend;
```

- Как его вызвать, чтобы в первый параметр попали столбцы employee_name country и city, а во второй – значения San Diego и Sydney ?

Задания

5) Есть шаг data, который использует библиотеку ecprg1.

- Напишите макрос, который сначала проверяет, есть ли такая библиотека в текущем сеансе
- Совет – воспользуйтесь функцией Libref
- Если она (библиотека) есть, шаг data выполняется
- Дальше проверяем код возврата шага data (может быть там были какие-то ошибки?)
- Если библиотеки в сеансе нет, нужно вывести в log сообщение об ошибке (совет – начните его со слова “ERROR: “ чтобы строчка в log выделилась красным)
- Если на шаге data возникли ошибки, вывести ещё одно сообщение в log

Задания

- Срок сдачи - до 23:59:59 31 октября 2014 г
  (праздник halloween)
- Программы присылать на адрес
  Dmitry.Zvezhinsky@sas.com
- Смотреть результаты на странице с лекциями:
  http://tiny.cc/msu_sas