Программирование на языке SAS BASE

Лекция 2. Основы.

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

Замечания об ошибках и опечатках просьба направлять лектору.

Циклы DO

Вывести в набор данных целые цифры от 1 до 5 (включительно)

1.

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

По умолчанию новая переменная инициализируется пропущенным значением

В данном случае – то же, что n=n+1;

Условие проверяется в конце цикла

2.

```
data test;
n=0;
   do while(n<5);
      n+1;
          output;
   end;
run;
```

Условие проверяется в начале цикла

3.

```
data test;
   do n=1 to 5;
          output;
   end;
run;
```

А если хочется, можно вообще без циклов:

```
data test;
n=1;
  lbl1:
  output;
  n=n+1;
  if n<=5 then go to lbl1;
run;
```

Особенности шага Data

- Посмотрим две программы:

№1	№2

data test;
  set work.test1;
run;
	data test;
	  set work.test1;
	  set work.test1;
	run;

Work.test

a	b
1	PRIVET
2	MIR!

- Вопросы: а) когда закончится выполнение программ?
- б) сколько итераций сделает цикл в каждой программе?
- в) будет ли отличаться результат?

Смотрим в log:

```
NOTE: There were 2 observations read from the data set WORK.TEST1.
NOTE: The data set WORK.TEST has 2 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
...
NOTE: There were 2 observations read from the data set WORK.TEST1.
NOTE: There were 2 observations read from the data set WORK.TEST1.
NOTE: The data set WORK.TEST has 2 observations and 2 variables.
```

Первый шаг data

Второй шаг data

Особенности шага Data

- Как работает эта программа?

```
data test;
  set work.test1;
  set work.test1;
run;
```

Итерация 1.
Первый set: записывает первое наблюдение test1 в PDV
Второй set: записывает первое наблюдение test1 в PDV
Неявный оператор output (Содержимое PDV переносится в новый набор данных)
Неявный оператор return
Итерация 2.
Первый set: записывает второе наблюдение test1 в PDV
Второй set: записывает второе наблюдение test1 в PDV
Неявный оператор output (Содержимое PDV переносится в новый набор данных)
Неявный оператор return
Итерация 3.
Первый set: наткнулись на конец файла, шаг data закончен.

Оператор put выводит в log содержимое одной или нескольких переменных PDV

Набегающие суммы

- Пример: каким образом пронумеровать наблюдения?

```
data test;
   ctr=0;
   put ctr=;
   set ecprg1.contacts;
   ctr=ctr+1;
   put _all_;
run;
```

Выводит значение переменной в log

Просмотр содержимого PDV (в log)

Не работает

```
data test;
   retain ctr 0;
   set ecprg1.contacts;
   ctr=ctr+1;
run;
```

Создается новая переменная ctr, инициализируется нулём, её значение не будет сбрасываться в missing в начале каждой итерации шага DATA

```
data test;
   set ecprg1.contacts;
   ctr+1;
run;
```

Эта запись – аналог двух операторов в предыдущей программе (retain и увеличение счетчика)

Как можно создать набор данных?

- Как вам удобно! Например, с помощью шага DATA:

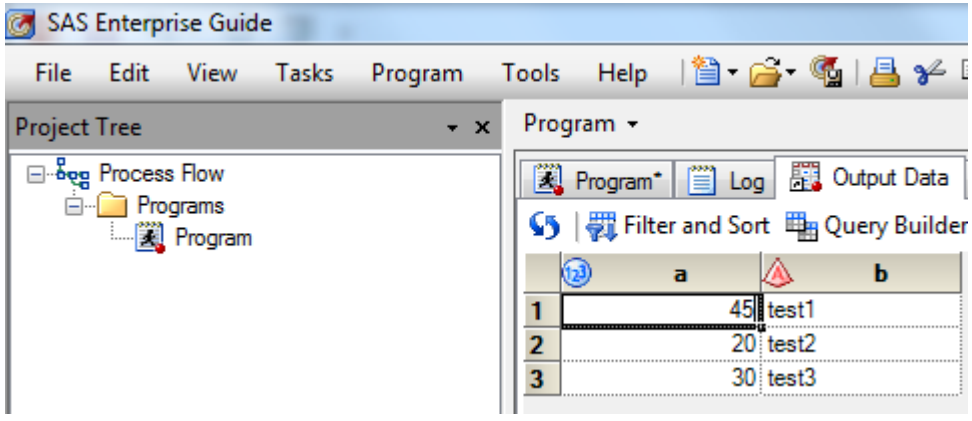
```
data test;
length b $ 10;
input a b $;
datalines;
10 test1
20 test2
30 test3
run;
proc print;run;
```

Создаём переменные в PDV
Оператор input – в какие переменные мы считываем содержимое неструктурированных данных?
После datalines идут сами данные (разделитель по умолчанию – пробел.

- Proc SQL (на следующей лекции):

```
proc sql;
create table work.test2 (a numeric, b char length=10);
insert into work.test2 set a=10, b="test1"
                    set a=20, b="test2"
                    set a=30, b="test3";
select * from work.test2;
quit;
```

А если бы мы пользовались не SAS U, то можно было бы заполнить таблицу в редакторе а-ля Эксель:



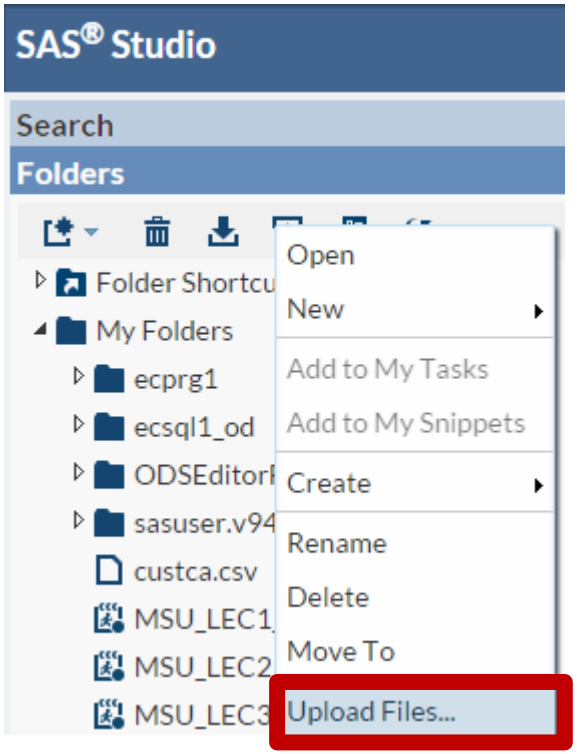
Импорт данных

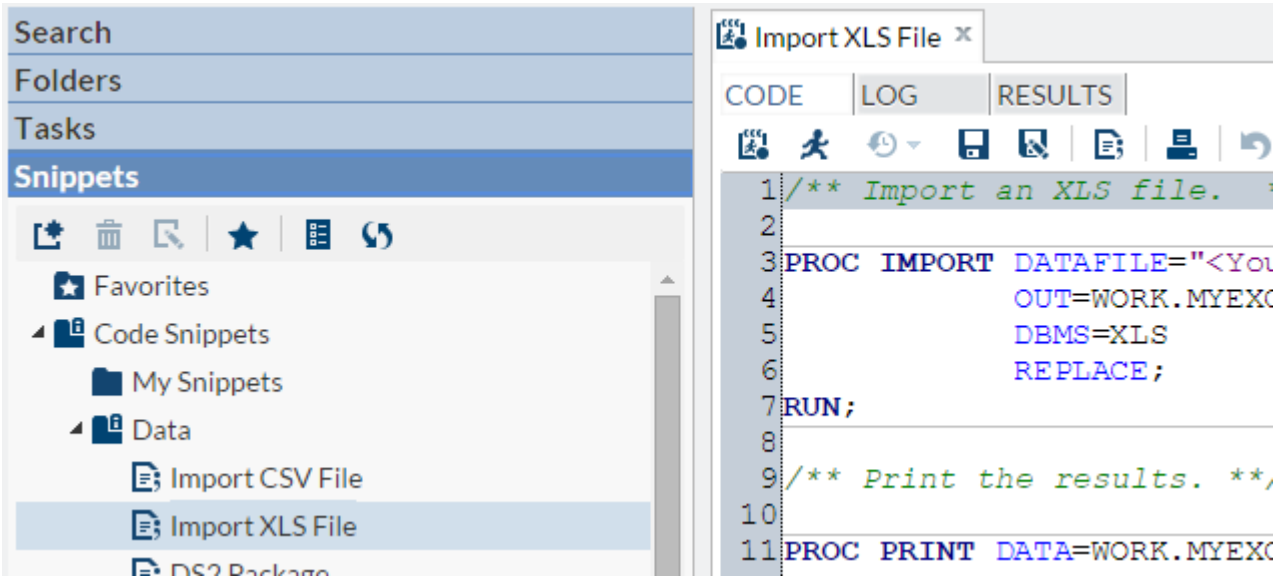
- А если данные уже лежат в xls, csv? Будем делать так:

1. Загружаем файл в My Folders любыми способами, например:

2. Code Snippets -> Data -> Import …
3. Меняем в программе строчку DATAFILE="~/имя_файла.xls”
4. Меняем название набора данных OUT=WORK.имя_набора_SAS
5. Запускаем.

Вместо WORK может быть ваша постоянная библиотека.





Сортировка наборов данных (Proc SORT)

```
proc sort data=ecprg1.contacts out=tempsort;
   by descending name;
run;
```

Куда будет выведен отсортированный набор данных?

По каким переменным сортировать?

Входящий набор данных

Proc SORT умеет искать и выделять дубликаты:

```
proc sort data=ecprg1.nonsalesdupes(keep=employee_id first last)
                                dupout=dup1 out=nondup1 NODUPKEY;
        by employee_id;
run;
proc sort data=ecprg1.nonsalesdupes(keep=employee_id first last)
                                dupout=dup2 out=nondup2 NODUPRECS;
        by employee_id;
run;
```

А

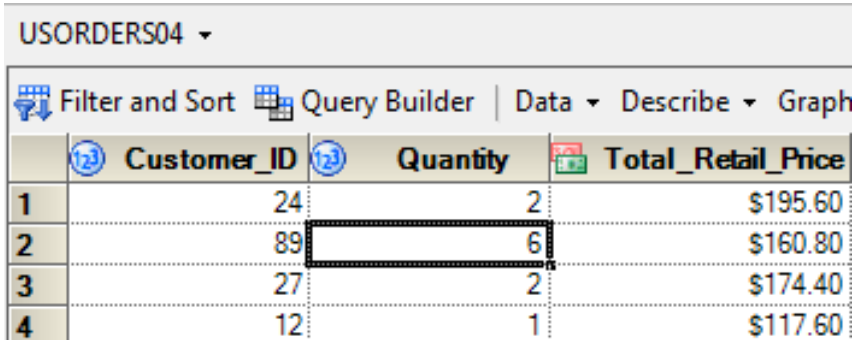
Б

А: Дубликатами считаются записи с одинаковым значением employee_id. Дубликаты выводятся в набор данных dup1, все остальные записи – в nondup1

Б: Дубликатами считаются записи, если они полностью совпадают друг с другом (все поля, которые мы читаем из nonsalesdupes: employee_id, first, last)

Оператор by (шаг DATA)

- Если набор данных отсортирован по некоторой переменной, на шаге DATA мы можем работать с маркерами начала и конца группы наблюдений с одним и тем же значением этой переменной.



	Customer_ID	Quantity	Total_Retail_Price
1	24	2	$195.60
2	89	6	$160.80
3	27	2	$174.40
4	12	1	$117.60

Пример:
Как посчитать сумму Quantity для каждого Customer_ID?

```
1  proc sort data=ecprg1.usorders04 out=usord_sort;
2          by customer_id;
3  run;
4
5  data summary_cust (keep=sum customer_id);
6     set usord_sort;
7     by customer_id;
8     if first.customer_id=1 then sum=0;
9     sum+quantity;
10    if last.customer_id then output;
11 run;
```

Сортировка!

First.*variable* = 1 для первого наблюдения в группе, =0 для других наблюдений,
Last.*variable* =1 для последнего наблюдения в группе, =0 для других наблюдений

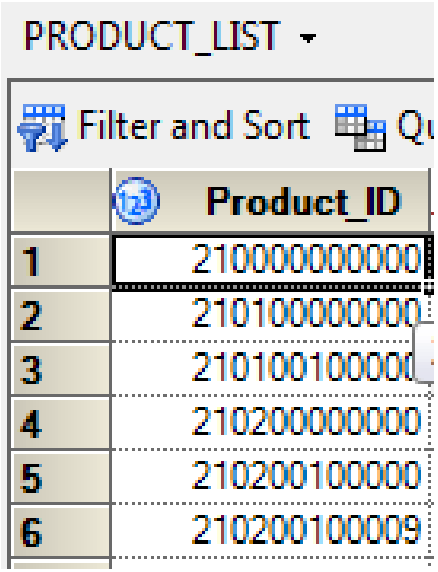
Операторы where, if (создание выборки на шаге DATA)

- Оператор where (как и опция набора данных where) работает только с переменными, которые уже есть во входящем наборе данных.
- Оператор where работает не только на шаге DATA, но и в процедурах.
- If работает со всеми переменными, которые присутствуют в PDV, в том числе со служебными и «новыми»

Задача: выбрать наблюдения, где product_list начинается с цифр 21

```
data plist1;
  set ecprg1.product_list;
  where product_id between 210000000000 and 219999999999 ;
run;

data plist2;
  set ecprg1.product_list;
  two_digits=int(product_id/10000000000);
  if two_digits = 21 ;
run;
```



В первом случае мы используем для фильтрации оператор where и переменную, которая уже есть во входящем наборе данных. Во втором – создаем новую переменную и пользуемся оператором “if” (без “then”, т.е. создающим выборку).

Операторы where, if (создание выборки на шаге DATA)

Полезные операторы и функции для работы с текстом:

- Поиск подстроки с учетом регистра

```
where string CONTAINS ‘Woman‘ ;
```

- Простой шаблон: знак подчеркивания – любой символ, процент – любое количество любых символов, регистр важен

```
where string LIKE ‘%Wom_n%‘ ;
```

- Поиск без учета регистра: см. функцию find

```
where find(string,‘woman‘, ‘I‘) >0 ;
```

- Вырезать подстроку: см. функцию substr, второй аргумент – номер символа, начиная с которого её вырезать, третий – сколько символов вырезать

```
where substr(string,1,5)= ‘woman‘ ;
```

  Эту же функцию можно использовать для замены части текста

- Длина текстовой строки без учета конечных пробелов: см. функцию length

```
where length(string)= 5 ;
```

- Замена в тексте комбинации символов: см. функцию tranwrd

Использование RegExp*

В SAS можно использовать «регулярные выражения» - стандарт для поиска шаблонов в тексте.

PRODUCT_LIST

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

	Product_ID	Product_Name	Supplier_ID
67	220100100421	Trois-fit Running Qtr Socks...	1303
68	220100100513	Woman's Deception Dress	1303
69	220100100516	Woman's Dri Fit Airborne T...	1303
70	220100100523	Woman's Dri-Fit Scoop Nec...	1303
71	220100100530	Woman's Emblished Work-...	1303
72	220100100536	Woman's Foxhole Jacket	1303

Примеры:

1) Поиск наблюдений, где product_name начинается со слова “Woman”:

```
data plist;
  set ecprg1.product_list;
  where prxmatch('/^Woman/', product_name )>0;
run;
```

Prxmatch возвращает номер первого символа, где найден указанный шаблон.

2) Поиск наблюдений и замена с помощью регулярных выражений:

```
data plist(keep=product_name product_name1);
  set ecprg1.product_list;
  where prxmatch('/Woman/', product_name )>0;
  product_name1= prxchange('s/Woman/Man/', -1, product_name);
run;
```

Prxchange производит указанное кол-во замен (-1 = без ограничений) по шаблону в тексте, указанном в третьем аргументе (product_name).

* http://support.sas.com/rnd/base/datastep/perl_regexp/regexp-tip-sheet.pdf

Call routines* (процедуры)

- Это ещё один вариант подпрограмм, отличается от функций тем, как возвращаются результаты.
- Их нужно вызывать с помощью оператора CALL:

  CALL routine-name (argument-1<, ...argument-n>);

  CALL routine-name (OF variable-list);

При этом аргумент может передавать значение в процедуру и/или в него возвращается какой-то результат работы процедуры.

Примеры использования шаблонов для названия переменных-аргументов (между прочим, это работает и в функциях):

```
call cats(result,of y1-y15);
call cats(result,of y:);
```

Первый пример - конкатенация 15 символьных переменных (y1, y2, y3, … , y 15) в переменную result. Второй пример – конкатенация всех переменных, которые начинаются с «y». (тогда они все должны быть правильного типа!)

```
call missing(sales);
```

Одному или нескольким аргументам будет присвоено пропущенное значение (missing) – числовое или символьное, в зависимости от типа аргумента.

* Справка: SAS(R) 9.3 Functions and CALL Routines: Reference

Очередь для переменной (Lag)

- Шаг DATA последовательно читает наблюдения из набора данных. Что делать, если нужно запомнить значение переменной, и использовать его на следующей итерации шага DATA?

Пример:Посчитаем разницу между значениями переменной hires в текущем и в прошлом году.

```
data dif1;
 set ecprg1.yearly_saleshires;
 dif0=lag(hires);
 dif1=hires-lag(hires);
 dif2=dif(hires);
run;
```

	Year	Hires	dif0	dif1	dif2
1	1974	23	.	.	.
2	1975	2	23	-21	-21
3	1976	4	2	2	2
4	1977	3	4	-1	-1
5	1978	7	3	4	4
6	1979	3	7	-4	-4
7	1980	4	3	1	1
8	1981	1	4	-3	-3

1. Можно смотреть не на одно, а на несколько наблюдений назад (функции lag2(..), lag3(..), …
2. Каждая функция lag в программе формирует свою очередь.
3. При вызове функции возвращается значение из начала очереди, далее происходит сдвиг и в конец очереди помещается текущее значение аргумента.
4. Перед первой итерацией шага DATA очередь заполняется пропущенными значениями (missing).

Массивы (шаг DATA)

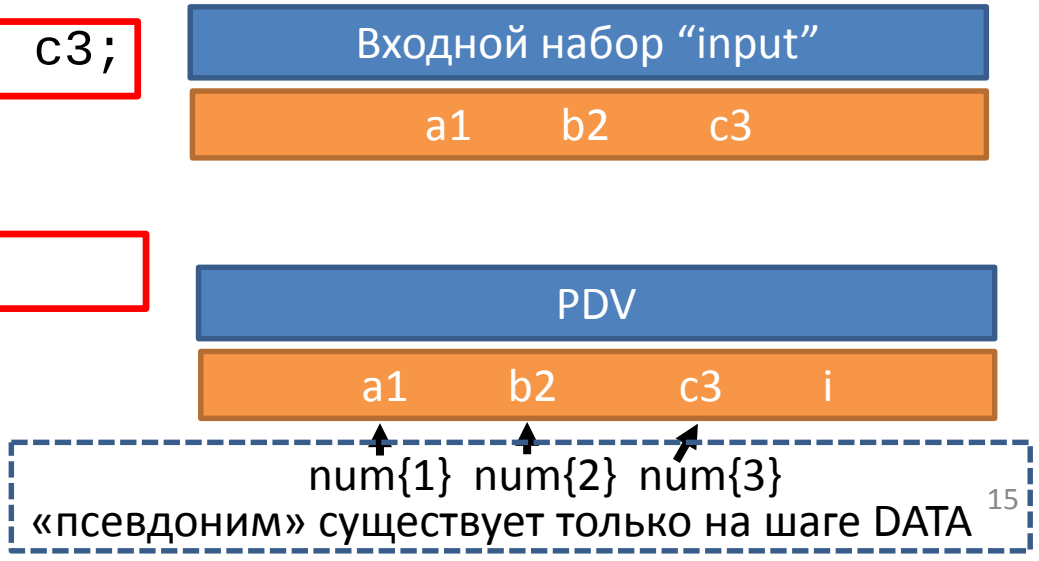
- Это – удобный способ автоматизировать обработку множества (однотипных) переменных.
- Также можно использовать для создания новых переменных.

Пример:

Есть исходный набор данных с числовыми переменными a1, b2, c3. Нужно к этим переменным прибавить 1.

Создадим «псевдоним с целочисленным индексом» для этих переменных, и используем цикл для повторяющихся действий.

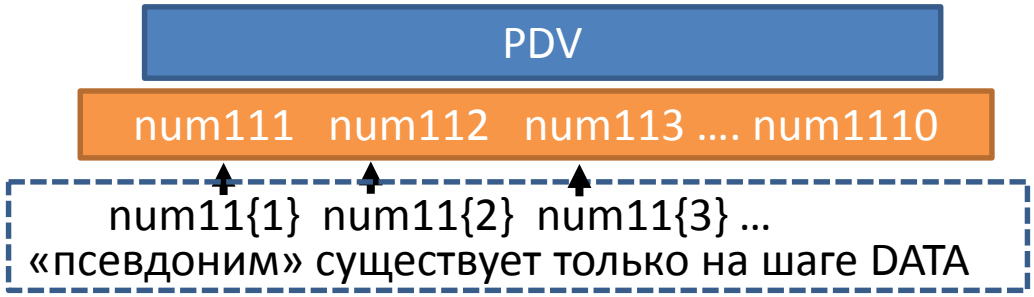
```
data temp1;
  array num {*} a1 b2 c3;
  set input;
  do i=1 to 3;
    num{i}=num{i}+1;
  end;
run;
```



Массивы (шаг DATA)

Создание массива из новых переменных

```
data test;
  array num11{10};
  num11{1}=34;
  num112=12;
run;
```



Число элементов массива,
элементы нумеруются с единицы

Начало названия новой переменной – как у массива, а затем идет число. К переменной можно обращаться и

по названию, которое создалось автоматически, и

по псевдониму – названию массива.

VIEWTABLE: Work.Test

	num111	num112	num113	num114	num115
1	34	12	.	.	

Массивы (шаг DATA)

- Массивы из символьных переменных.

```
data test;
   array char{6} $ 20;
run;
```

Массив из 6 переменных символьного типа, которые имеют размер 20 байт

```
data test1;
   array char{*} $ 20 var22-var33;
run;
```

Размер массива может быть опущен (заменён символом «*»), если он ясен «из контекста», то есть в определении можно понять, для скольких переменных нужно создать «псевдоним». В названии переменных для хранения данных, на которые «ссылается» массив, может быть применён шаблон «-», который понимает целое число в конце названия переменной как счетчик.

Переменные могут быть взяты из набора данных, или созданы на шаге DATA

```
data test2;
   array char{*} $ 20 aa bb cc;
   set input;
run;
```

Массивы (шаг DATA)

- Врèменные массивы

Массив можно использовать не только для обращения к переменным в PDV и их создания, но и как средство для выделения области памяти, которая не относится к PDV.

```
data test1;
   array char{10} $ 20 _temporary_;
run;
```

Особенности:

1) Обращение к элементам массива – только через название массива и индекс элемента.
2) В PDV не создаётся никаких переменных для хранения данных.

```
data test1;
   array char{10} $ 20 _temporary_;
   put _all_;
run;
```

```
 _ERROR_=0 _N_=1
 NOTE: The data set WORK.TEST1 has 1 observations and 0 variables.
 NOTE: DATA statement used (Total process time):
      real time          0.00 seconds
      cpu time           0.01 seconds
```

Массивы: особенности

- Особенности массивов в SAS

1) Память для массива выделяется на фазе компиляции шага DATA и освобождается при его завершении
2) На шаге DATA нет никаких инструментов для динамического выделения памяти (“malloc”) и для её освобождения (“free”), то есть память нельзя выделить позже, чем начнётся шаг DATA, и освободить раньше, чем он завершится.
3) Размер массива не может быть динамическим. Он должен быть известен на момент компиляции шага DATA.

Массивы: ошибки

1) выход за границы массива

```
62    data test1;
63       array char{10} $ 20 _temporary_;
64       char{30}="ggg";
65    run;

ERROR: Array subscript out of range at line 64 column 4.
_ERROR_=1 _N_=1
NOTE: The SAS System stopped processing this step because of errors.
WARNING: The data set WORK.TEST1 may be incomplete.  When this step was stopped
         there were 0 observations and 0 variables.
WARNING: Data set WORK.TEST1 was not replaced because this step was stopped.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

2) смешивание типов

```
66    data test1;
67       array char{2} aa bb;
68       aa=10;
69       bb="ggg";
70    run;

NOTE: Character values have been converted to numeric
      values at the places given by: (Line):(Column).
      69:7
NOTE: Invalid numeric data, 'ggg' , at line 69 column 7.
aa=10 bb=. _ERROR_=1 _N_=1
NOTE: The data set WORK.TEST1 has 1 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.05 seconds
      cpu time            0.00 seconds
```

Переменная bb числового типа была создана с помощью оператора array.

Массивы: ошибки

- Ошибки

2) смешивание типов

Пытаемся создать текстовый массив из набора данных, который уже содержит числовые переменные.

```
83   data test1;
84      array char{2}  $ 10 aa bb;
85      set numeric;
ERROR: Variable aa has been defined as both character and numeric.
ERROR: Variable bb has been defined as both character and numeric.
86   run;
```

3) нецелые индексы (у SAS нет специального целого типа) – у индекса отбрасывается дробная часть.

```
data test2;
   array num1{*} aa bb cc;
   aa=1;
   bb=2;
   cc=3;
   put num1{1.9}; %в log появится «1».
run;
```

Массивы: рецепты

- Что можно делать с массивами? (иллюстрации)

1) Вычисление функций-«агрегатов» по числовому массиву

```
data test;
  a1=1;
  b2=2;
  c3=3;
  d4=4;
  array eee {4} a1 b2 c3 d4;
  a=max(of eee[*]);
  put a=;
run;
```

2) Проверка на вхождение значения в массив

```
data _null_;
  array list {3} $ 20 ("alpha" "beta" "dima");
  string="dima";
  if string in list then put "Ok!";
run;
```

MAX finds maximum value in list.
MEDIAN find median of a list of values.
CALL MISSING sets all values in as missing.
Example: CALL MISSING(OF T(*));
CALL SORTN sorts values in the array

http://support.sas.com/resources/papers/97529_Using_Arrays_in_SAS_Programming.pdf
http://www.stattutorials.com/SAS/Tutorial-SAS-ARRAYS.html

Массивы: особенности

- Обратите внимание, что элементы массива (будучи обычными переменными PDV) могут быть сброшены в missing
- Если переменная на шаге DATA не была взята из существующего набора данных, то она будет сброшена на последующих итерациях шага DATA.

```
data test;
do i=1 to 10;
  output;
end;
run;
data _null_;
  set test;
  array my{3} a b c;
  if _n_=1 then do;
          a=1;b=2;c=3;
  end;
  put _all_;
run;
```

```
i=1 a=1 b=2 c=3 _ERROR_=0 _N_=1
i=2 a=. b=. c=. _ERROR_=0 _N_=2
i=3 a=. b=. c=. _ERROR_=0 _N_=3
i=4 a=. b=. c=. _ERROR_=0 _N_=4
i=5 a=. b=. c=. _ERROR_=0 _N_=5
i=6 a=. b=. c=. _ERROR_=0 _N_=6
i=7 a=. b=. c=. _ERROR_=0 _N_=7
i=8 a=. b=. c=. _ERROR_=0 _N_=8
i=9 a=. b=. c=. _ERROR_=0 _N_=9
i=10 a=. b=. c=. _ERROR_=0 _N_=10
```

- Если вы инициализируете массив, то его значения на последующих итерациях сброшены не будут (аналог применения оператора retain).

PROC PRINT

- Распечатка набора данных (листинг) в виде отчета.

PROC PRINT <option(s)>;

BY <DESCENDING> variable-1 <…<DESCENDING>variable-n>;

….

VAR variable(s) <option>;

В заголовке PROC Print указывается название набора данных и некоторые настройки отчета.

Оператор By (не обязательный) задает разбивку отчета по указанным переменным (требуется предварительная сортировка набора данных)

Оператор Var (не обязательный) задает структуру отчета, в нем перечисляются переменные, которые мы хотим вывести в отчет. Если этого оператора нет – выводятся все столбцы из набора данных.

Пример:

```
proc print data= ecprg1.customer noobs;
var customer_id country gender;
run;
```

*** Base SAS® 9.3 Procedures Guide -> PRINT Procedure**

PROC MEANS

- Отчет с описательной статистикой для набора данных (в том числе с разбивкой по классифицирующей переменной)

```
proc means data=ecprg1.monthly_prices n mean max min;
  var unit_cost_price;
  class month;
run;
```

MONTHLY_PRICES

Filter and Sort | Query Builder | Data

	Month	Unit_Cost_Price
1	6	$15.50
2	1	$17.80
3	1	$9.25
4	2	$9.30
5	4	$9.00
6	4	$2.30

The MEANS Procedure

Analysis Variable : Unit_Cost_Price Unit Cost Price					
Month	N Obs	N	Mean	Maximum	Minimum
1	17	17	34.0000000	131.5000000	3.3000000
2	22	22	36.9295455	124.9000000	9.3000000
3	46	46	49.5728261	289.9500000	4.1000000
4	32	32	56.8000000	253.2000000	2.3000000
5	35	35	60.7985714	251.3500000	8.4500000
6	19	19	52.5973684	315.1500000	7.5000000

Page Break

Процедура умеет рассчитывать ~ 30 разных статистик для выборки

*** Base SAS® 9.3 Procedures Guide -> MEANS Procedure**

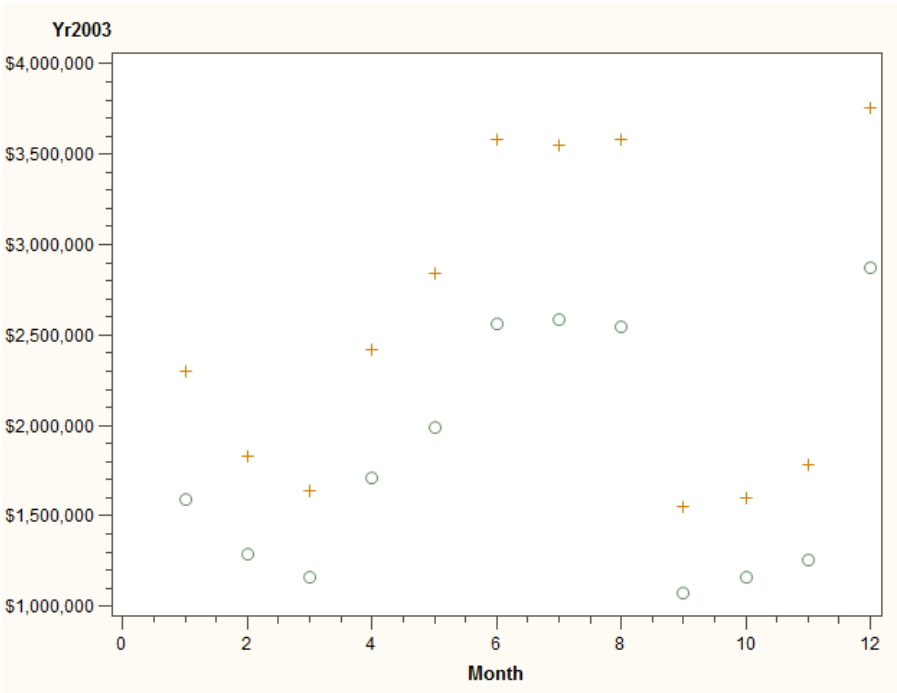
PROC SGPLOT

Пример: построить два линейных графика на одних осях координат, данные находятся в наборе данных budget.

BUDGET

Filter and Sort | Query Builder | Data | Describe | Gra

	Month	Yr2003	Yr2004
1	1	$1,590,000	$1,880,000
2	2	$1,290,000	$1,550,000
3	3	$1,160,000	$1,380,000
4	4	$1,710,000	$2,100,000
5	5	$1,990,000	$2,350,000
6	6	$2,560,000	$3,020,000




```
proc sgplot data=ecprg1.budget;
  scatter y=yr2003 x=month ;
  scatter y=yr2005 x=month ;
run;
```

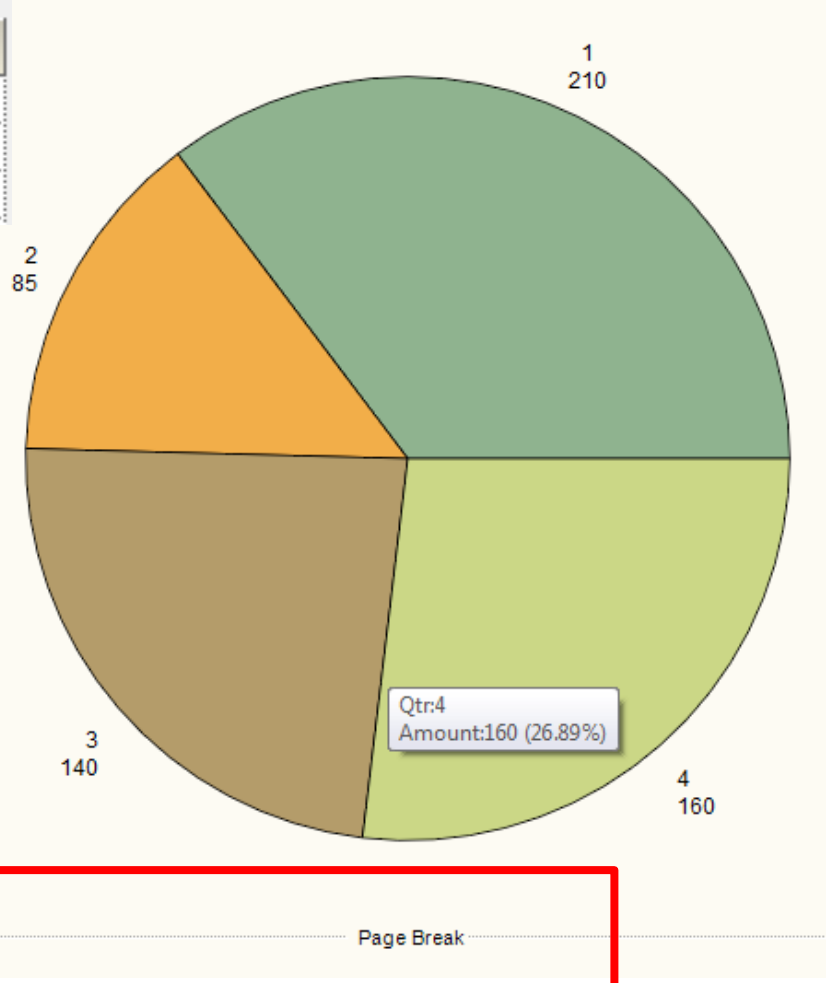
SAS® 9.3 ODS Graphics Procedures Guide, Third Edition

Диаграммы (нет в SAS U)

DONATE

Filter and Sort | Query Builder | Data | Describe | G

	Employee_ID	Qtr	Amount
1	11036	2	20
2	11036	3	25
3	11057	1	10




```
proc gchart data=ecprg1.donate;
   pie qtr /
      DISCRETE sumvar=amount;
run;
quit;
```

На что способен SAS/Graph (нет в SAS U)
(примеры + программы)
http://robslink.com/SAS/Home.htm
http://support.sas.com/sassamples/graphgallery/**PROC_GCHART**.html

Манипуляции с выводом

1. Сохранение нескольких отчетов в файл стороннего формата

```
ods pdf file='~/my.pdf';
proc print data=ecprg1.donate;
run;
ods pdf close;
```

Пояснения:

*) ~/ : обозначение домашней директории в linux

*) Ищите полученный pdf в общей директории, подключенной к виртуальной машине (см. log):

```
44 ods pdf file='~/my.pdf';
NOTE: Writing ODS PDF output to DISK destination "/folders/myfolders/my.pdf", printer "PDF".
```

*) Вместо формата pdf можно использовать другие форматы: HTML, RTF, … LaTeX,…

*) Справка: **SAS(R) 9.3 Output Delivery System: User's Guide, Second Edition**

Манипуляции с выводом

2. Сохранение объекта из отчета в набор данных

```
proc means data=ecprg1.budget;
  var yr2003 yr2004;
run;
```

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Yr2003	12	1816666.67	667278.51	1070000.00	2870000.00
Yr2004	12	2087500.00	733647.49	1180000.00	3120000.00

Page Break

Шаг 1. Узнать имя объекта:

```
ods trace on;
proc means data=ecprg1.budget;
  var yr2003 yr2004;
run;
ods trace off;
```

Program

Program* | Log | Results

Export ▾ Send To ▾ Create ▾ | Project Log

```
Output Added:
-------------
Name:          Summary
Label:         Summary statistics
Template:      base.summary
Path:          Means.Summary
-------------
```

Подставить сюда

Шаг 2:

```
ods output Summary=work.tab1;
proc means data=ecprg1.budget;
  var yr2003 yr2004;
run;
```

Program* | Log | Output Data | Results

Filter and Sort | Query Builder | Data ▾ Describe ▾ Graph ▾ Analyz

	VName_Yr2003	Yr2003_N	Yr2003_Mean	Yr2003_StdDe
1	Yr2003	12	1816666.6667	667278.5071

Program* | Log | Output Data | Results

Export ▾ Send To ▾ Create ▾ | Project Log | Properties

```
NOTE: The data set WORK.TAB1 has 1 observations and 12 variables.
```

Задания

0) Установить SAS U по инструкции и настроить его.

- Самостоятельно проделать демонстрации со слайдов, подумать над вопросами

1) Создайте с помощью Excel файл с двумя колонками, x и y. Причем x=-10:0.1:10, а y вычисляется как sin(x)

- Импортируйте файл Excel в набор данных SAS
- Напишите шаг Data, где вычисляется «скользящая» сумма по этому набору данных с заданным «окном» (пусть 5 точек)
- Постройте значение этой суммы на графике

Задания

2) Напишите шаг data, где создаётся набор данных с двумя переменными (x, y) и 50 наблюдениями. В каждом наблюдении x выбирается случайно (равномерно распределено от -1 до 1), а y считается как x^2.

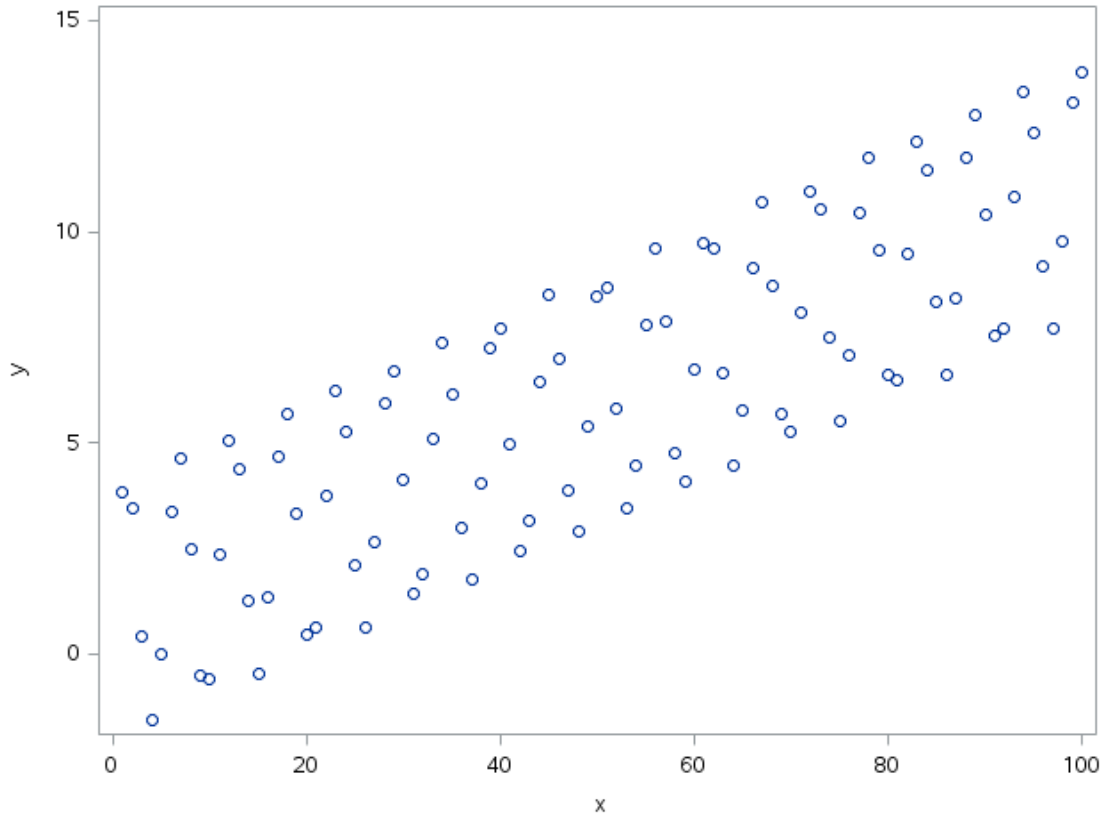
- Отсортируйте этот набор данных по переменной x.
- Численно проинтегрируйте y(x), например, с помощью формулы трапеций, воспользовавшись вторым шагом data.

3) Найдите самостоятельно «рецепты», как можно объединить «по вертикали» два набора данных. Напишите программу, которая это делает с таблицами ecprg1.emps2008, ecprg1.emps2009 и ecprg1.emps2010

Задания

4) С помощью шага DATA реализуйте медианный фильтр некоторой числовой последовательности, которая создаётся программой ниже. Постройте график исходных данных random и результата работы вашего медианного фильтра с окном 10 (точек).

```
data random;
do x=1 to 100;
 y=1+x/10+3*sin(x*20);
 output;
end;
run;
proc sgplot;
scatter x=x y=y;
run;
quit;
```



Задания

5) Как вы думаете, какая буква будет первой при сортировке русского алфавита по возрастанию?

- Попробуйте выполнить программу:

```
data russian;
input a $ 2. @@; /*помните про UTF-8?*/
if a ne ' ';
datalines;
АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЬЫЪЭЮЯ
;
run;
proc sort data=russian;by a;
proc print data=russian;run;
```

- Почему буква «А» не первая?
- Как же заставить proc sort сортировать русский алфавит в правильном порядке? Подсказка – посмотрите в документации опцию SORTSEQ= процедуры SORT.