Applied Statistics and Programming (ASAP)
Seminar Series- Session #24

September 2004

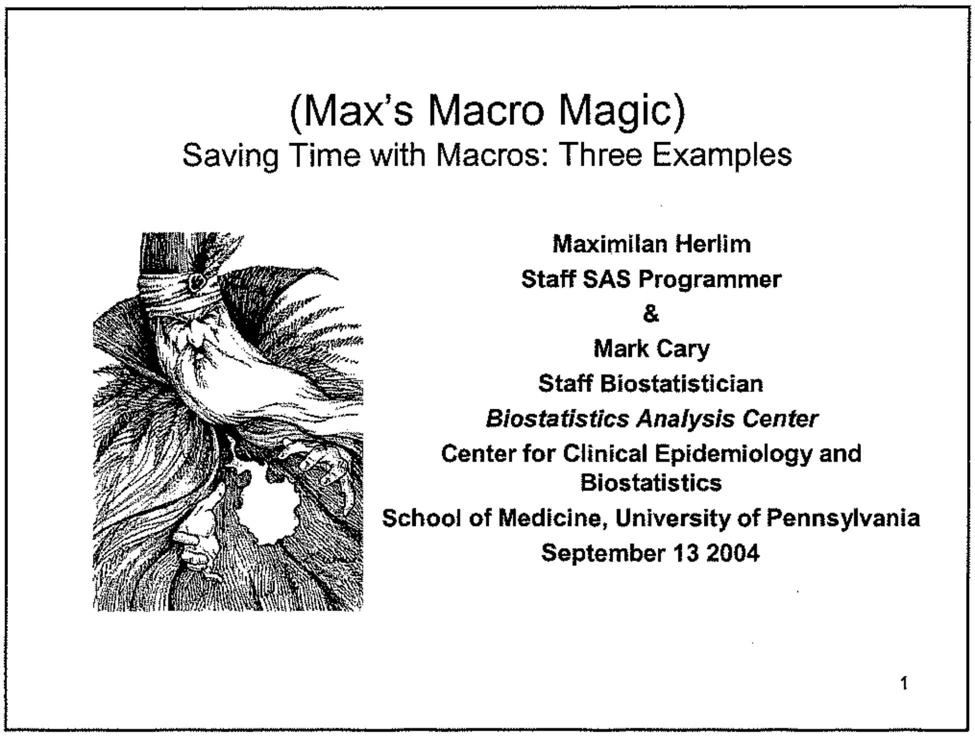
Saving Time with Macros: Three Examples

Presented by Maximilian Herlim and Mark S. Cary, PhD.

Biostatistics Analysis Center

Center for Clinical Epidemiology and Biostatistics

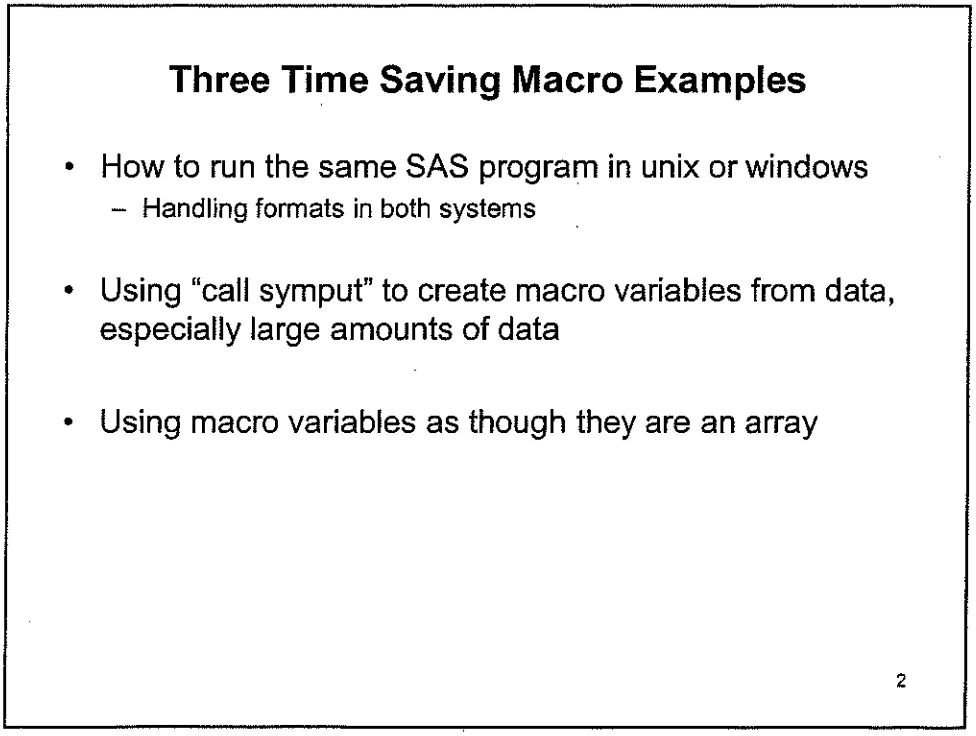
University of Pennsylvania School of Medicine



(Max's Macro Magic)

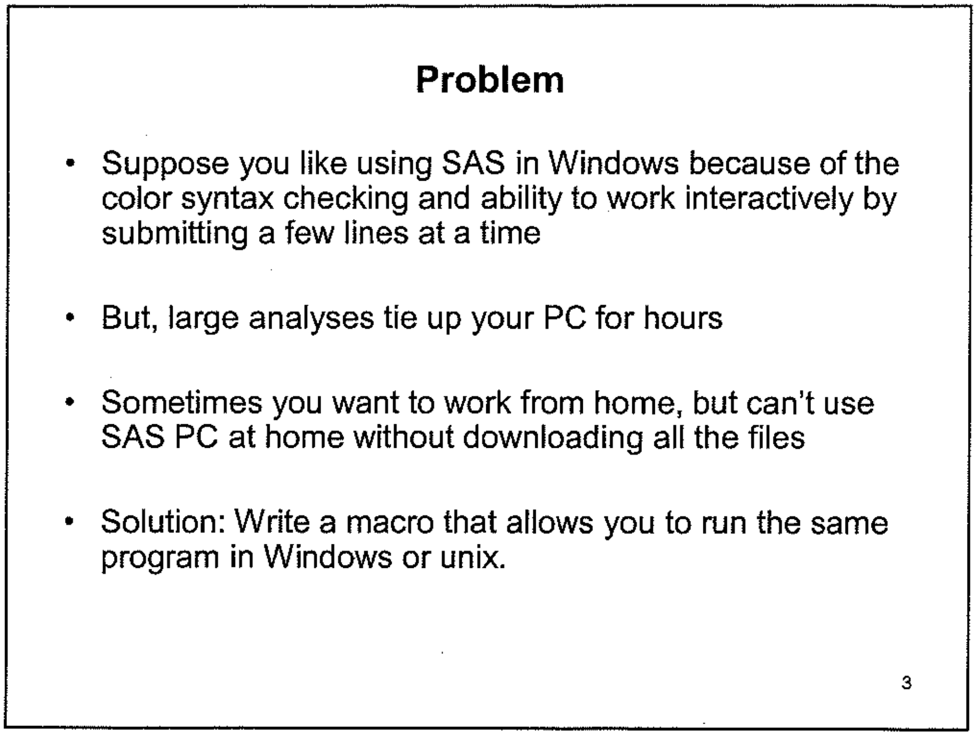
Saving Time with Macros: Three Examples

Maximilan Herlim
Staff SAS Programmer
&
Mark Cary
Staff Biostatistician
Biostatistics Analysis Center
Center for Clinical Epidemiology and Biostatistics
School of Medicine, University of Pennsylvania
September 13 2004



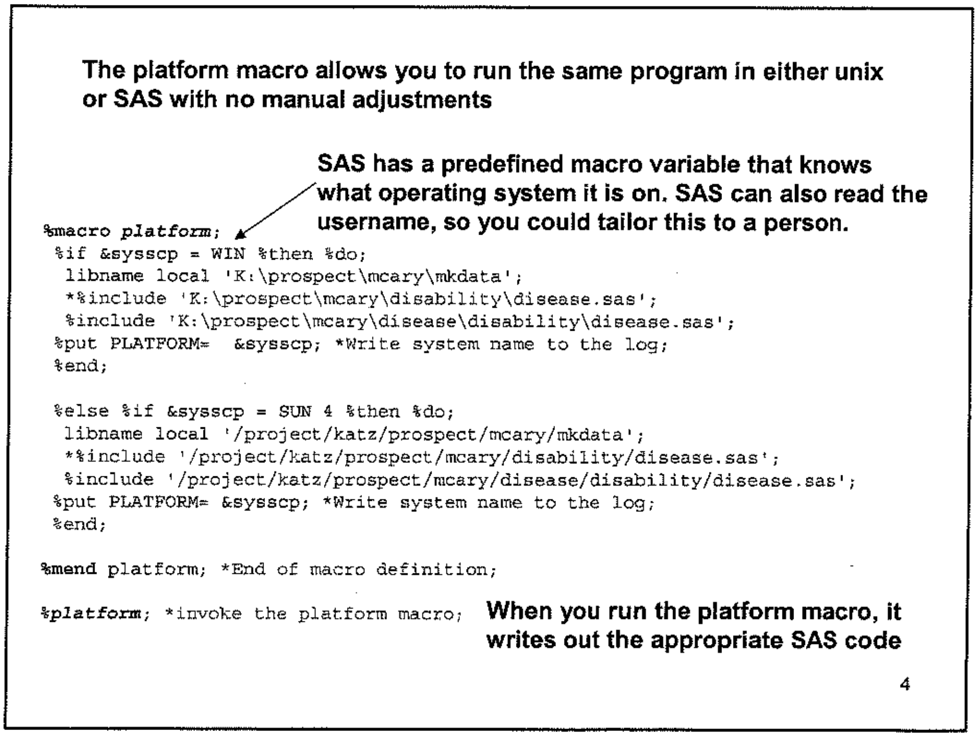
Three Time Saving Macro Examples

- How to run the same SAS program in unix or windows
  - Handling formats in both systems
- Using “call symput” to create macro variables from data, especially large amounts of data
- Using macro variables as though they are an array



Problem

- Suppose you like using SAS in Windows because of the color syntax checking and ability to work interactively by submitting a few lines at a time
- But, large analyses tie up your PC for hours
- Sometimes you want to work from home, but can't use SAS PC at home without downloading all the files
- Solution: Write a macro that allows you to run the same program in Windows or unix.

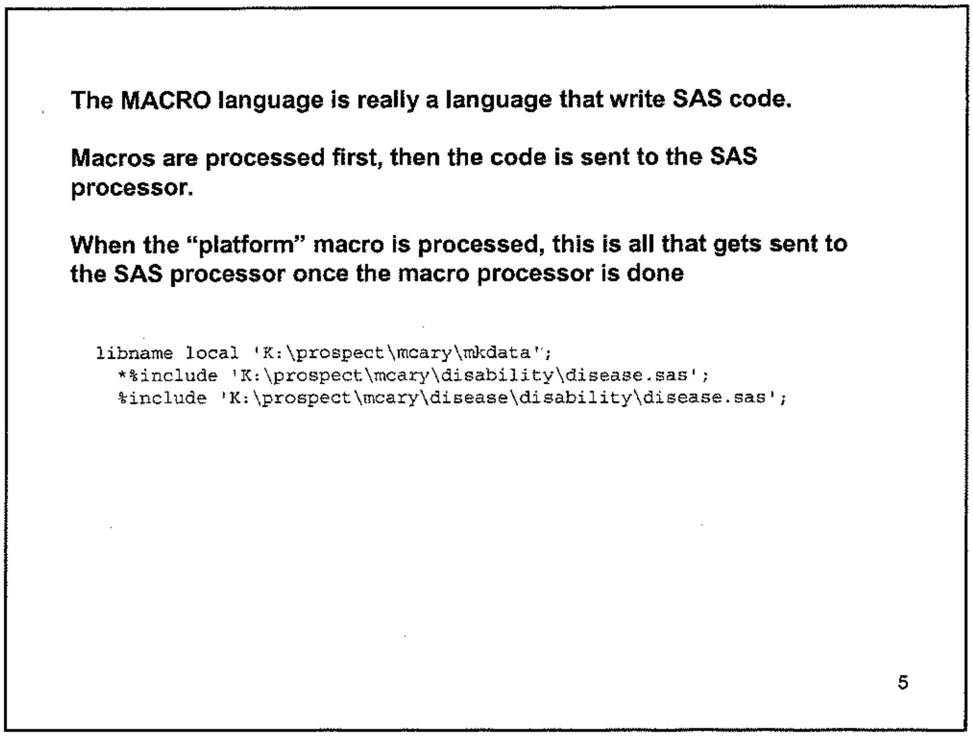


The platform macro allows you to run the same program in either unix or SAS with no manual adjustments

SAS has a predefined macro variable that knows what operating system it is on. SAS can also read the username, so you could tailor this to a person.

```
%macro platform;
 %if &sysscp = WIN %then %do;
  libname local 'K:\prospect\mcary\mkdata';
  *%include 'K:\prospect\mcary\disability\disease.sas';
  %include 'K:\prospect\mcary\disease\disability\disease.sas';
 %put PLATFORM=  &sysscp; *Write system name to the log;
 %end;

 %else %if &sysscp = SUN 4 %then %do;
  libname local '/project/katz/prospect/mcary/mkdata';
  *%include '/project/katz/prospect/mcary/disability/disease.sas';
  %include '/project/katz/prospect/mcary/disease/disability/disease.sas';
 %put PLATFORM= &sysscp; *Write system name to the log;
 %end;

%mend platform; *End of macro definition;

%platform; *invoke the platform macro;
```

When you run the platform macro, it writes out the appropriate SAS code



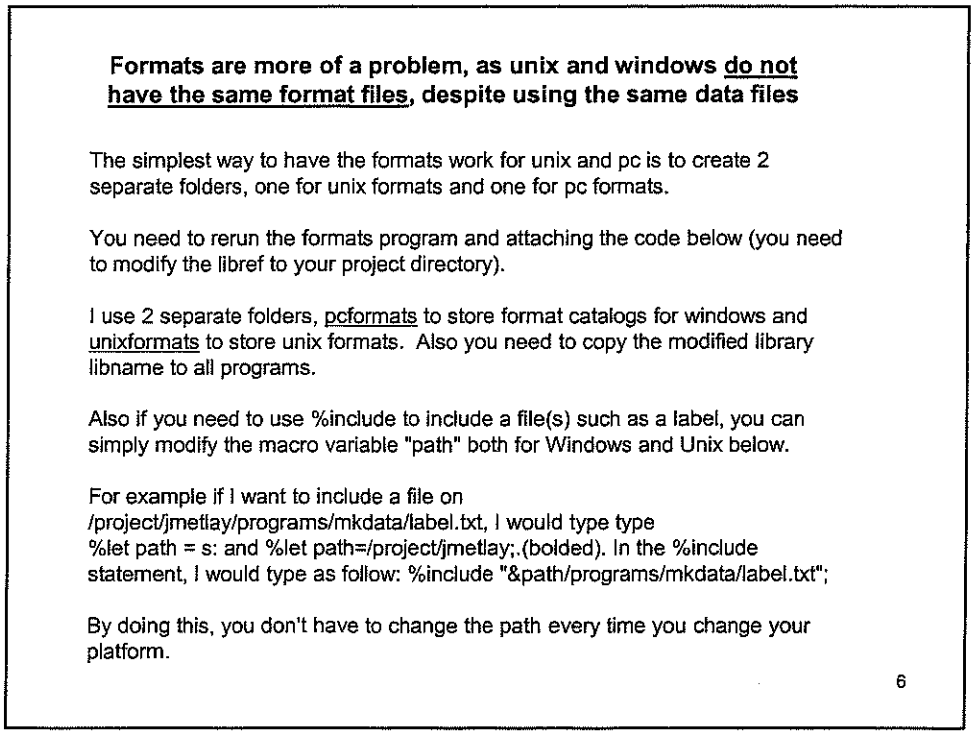
The MACRO language is really a language that write SAS code.

Macros are processed first, then the code is sent to the SAS processor.

When the “platform” macro is processed, this is all that gets sent to the SAS processor once the macro processor is done

```
libname local 'K:\prospect\mcary\mkdata';
  *%include 'K:\prospect\mcary\disability\disease.sas';
  %include 'K:\prospect\mcary\disease\disability\disease.sas';
```

5



Formats are more of a problem, as unix and windows <u>do not have the same format files</u>, despite using the same data files

The simplest way to have the formats work for unix and pc is to create 2 separate folders, one for unix formats and one for pc formats.

You need to rerun the formats program and attaching the code below (you need to modify the libref to your project directory).

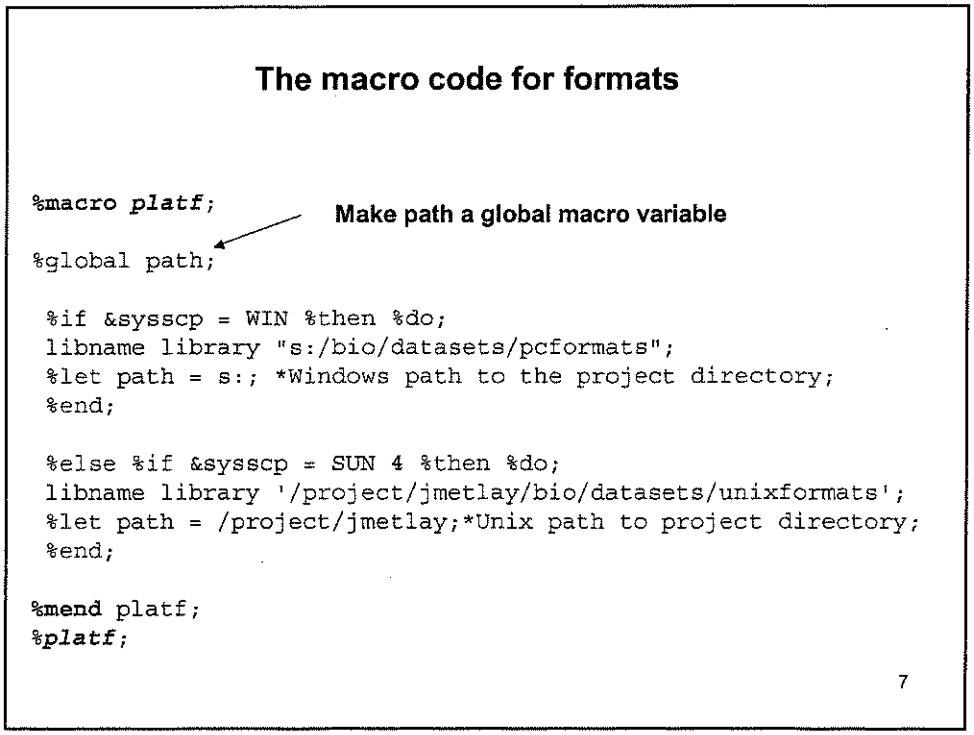
I use 2 separate folders, <u>pcformats</u> to store format catalogs for windows and <u>unixformats</u> to store unix formats. Also you need to copy the modified library libname to all programs.

Also if you need to use %include to include a file(s) such as a label, you can simply modify the macro variable "path" both for Windows and Unix below.

For example if I want to include a file on /project/jmetlay/programs/mkdata/label.txt, I would type type %let path = s: and %let path=/project/jmetlay;.(bolded). In the %include statement, I would type as follow: %include "&path/programs/mkdata/label.txt";

By doing this, you don't have to change the path every time you change your platform.

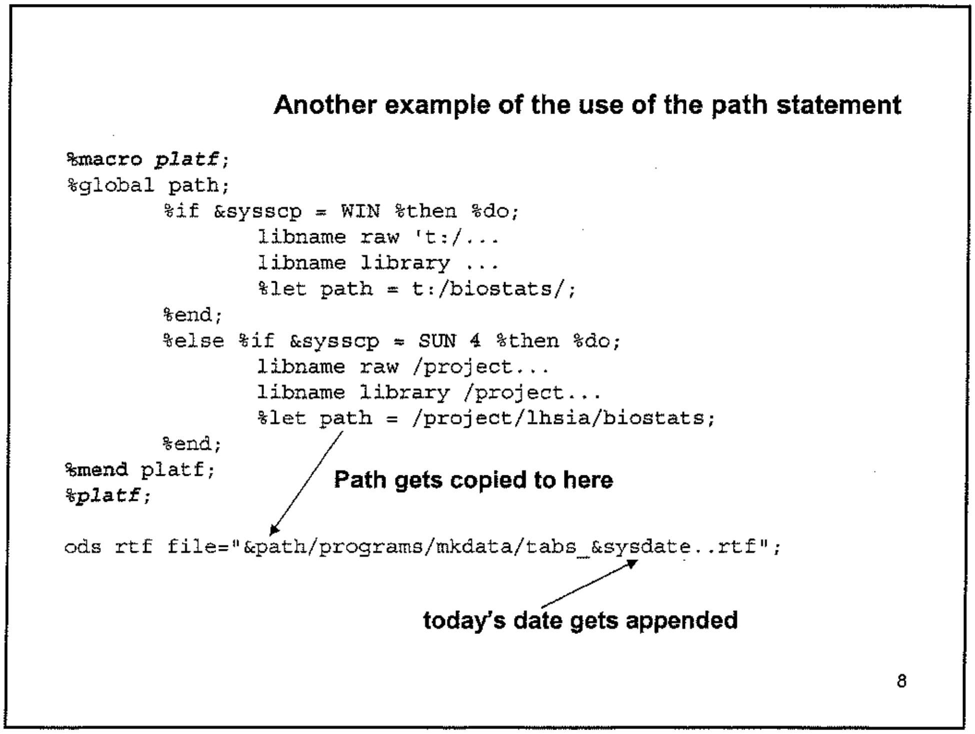
6



The macro code for formats

Make path a global macro variable

```
%macro platf;

%global path;

 %if &sysscp = WIN %then %do;
 libname library "s:/bio/datasets/pcformats";
 %let path = s:; *Windows path to the project directory;
 %end;

 %else %if &sysscp = SUN 4 %then %do;
 libname library '/project/jmetlay/bio/datasets/unixformats';
 %let path = /project/jmetlay;*Unix path to project directory;
 %end;

%mend platf;
%platf;
```



Another example of the use of the path statement

```
%macro platf;
%global path;
        %if &sysscp = WIN %then %do;
                libname raw 't:/...
                libname library ...
                %let path = t:/biostats/;
        %end;
        %else %if &sysscp = SUN 4 %then %do;
                libname raw /project...
                libname library /project...
                %let path = /project/lhsia/biostats;
        %end;
%mend platf;
%platf;

ods rtf file="&path/programs/mkdata/tabs_&sysdate..rtf";
```

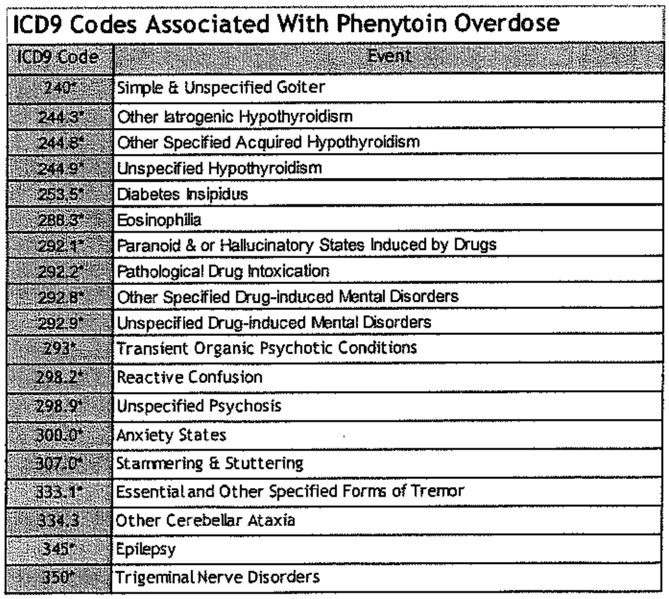
Path gets copied to here

today's date gets appended

Problem

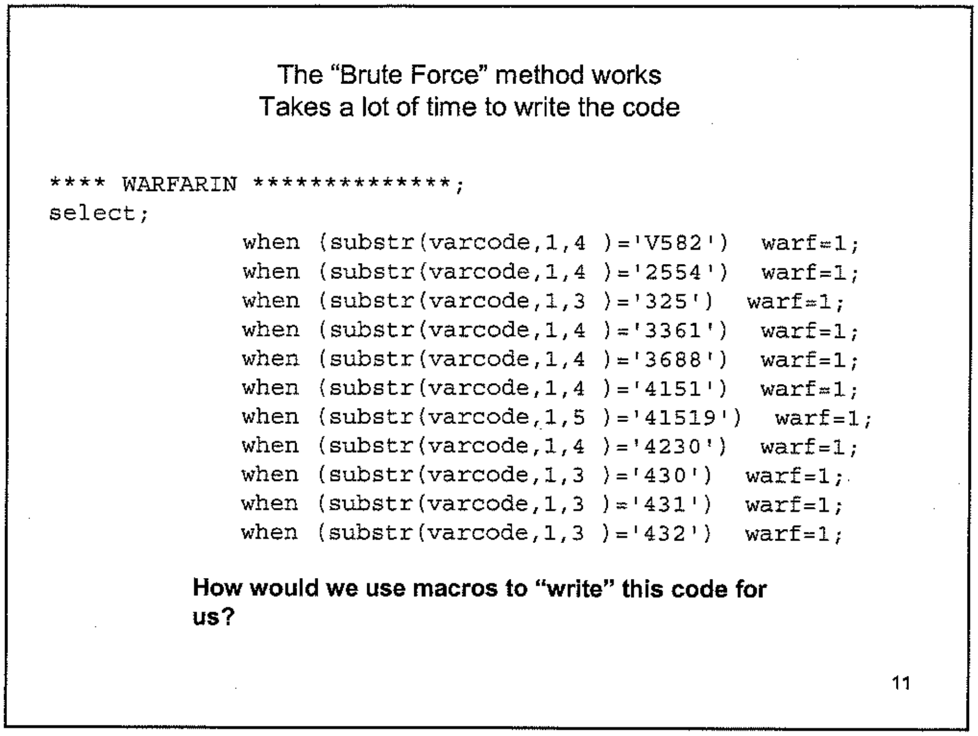
- You have to scan a number of diagnoses lists from a spreadsheet and write code to pick specific diagnoses
- You have hundreds of lines diagnoses
- **Solution**: Let the macro write the code

The Request--Find any hospitalizations that have any diagnoses that match those on these lists



ICD9 Codes Associated With Phenytoin Overdose

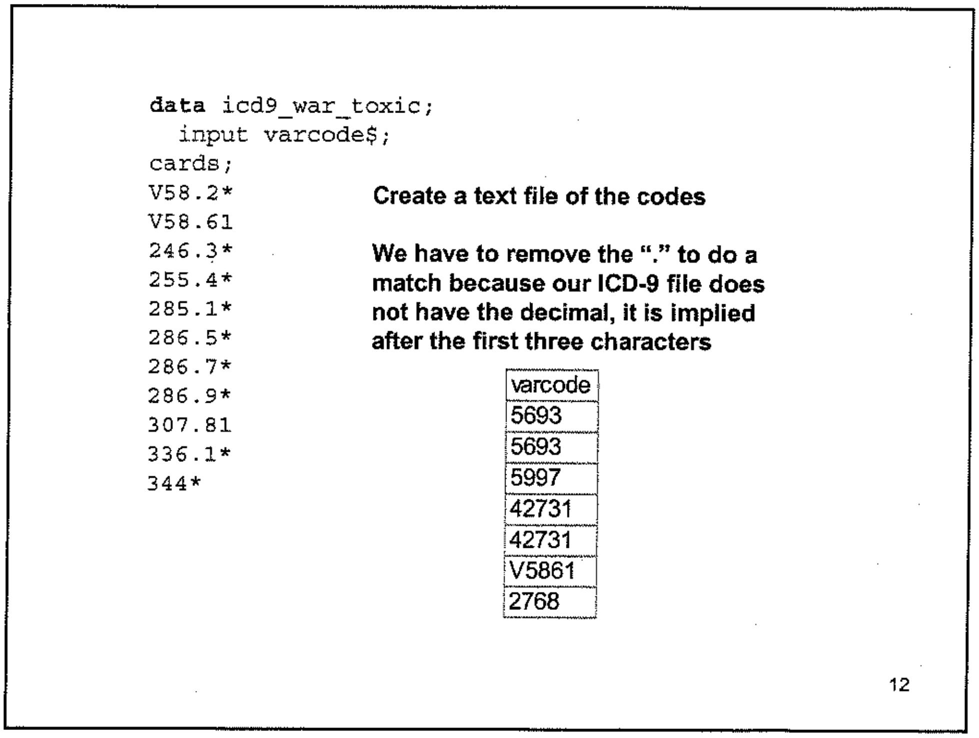
ICD9 Code	Event
240*	Simple & Unspecified Goiter
244.3*	Other latrogenic Hypothyroidism
244.8*	Other Specified Acquired Hypothyroidism
244.9*	Unspecified Hypothyroidism
253.5*	Diabetes Insipidus
288.3*	Eosinophilia
292.1*	Paranoid & or Hallucinatory States Induced by Drugs
292.2*	Pathological Drug Intoxication
292.8*	Other Specified Drug-induced Mental Disorders
292.9*	Unspecified Drug-induced Mental Disorders
293*	Transient Organic Psychotic Conditions
298.2*	Reactive Confusion
298.9*	Unspecified Psychosis
300.0*	Anxiety States
307.0*	Stammering & Stuttering
333.1*	Essential and Other Specified Forms of Tremor
334.3	Other Cerebellar Ataxia
345*	Epilepsy
350*	Trigeminal Nerve Disorders



The "Brute Force" method works
Takes a lot of time to write the code

```
**** WARFARIN **************;
select;
           when (substr(varcode,1,4 )='V582')  warf=1;
           when (substr(varcode,1,4 )='2554')  warf=1;
           when (substr(varcode,1,3 )='325')  warf=1;
           when (substr(varcode,1,4 )='3361')  warf=1;
           when (substr(varcode,1,4 )='3688')  warf=1;
           when (substr(varcode,1,4 )='4151')  warf=1;
           when (substr(varcode,1,5 )='41519')  warf=1;
           when (substr(varcode,1,4 )='4230')  warf=1;
           when (substr(varcode,1,3 )='430')  warf=1;
           when (substr(varcode,1,3 )='431')  warf=1;
           when (substr(varcode,1,3 )='432')  warf=1;
```

How would we use macros to "write" this code for us?

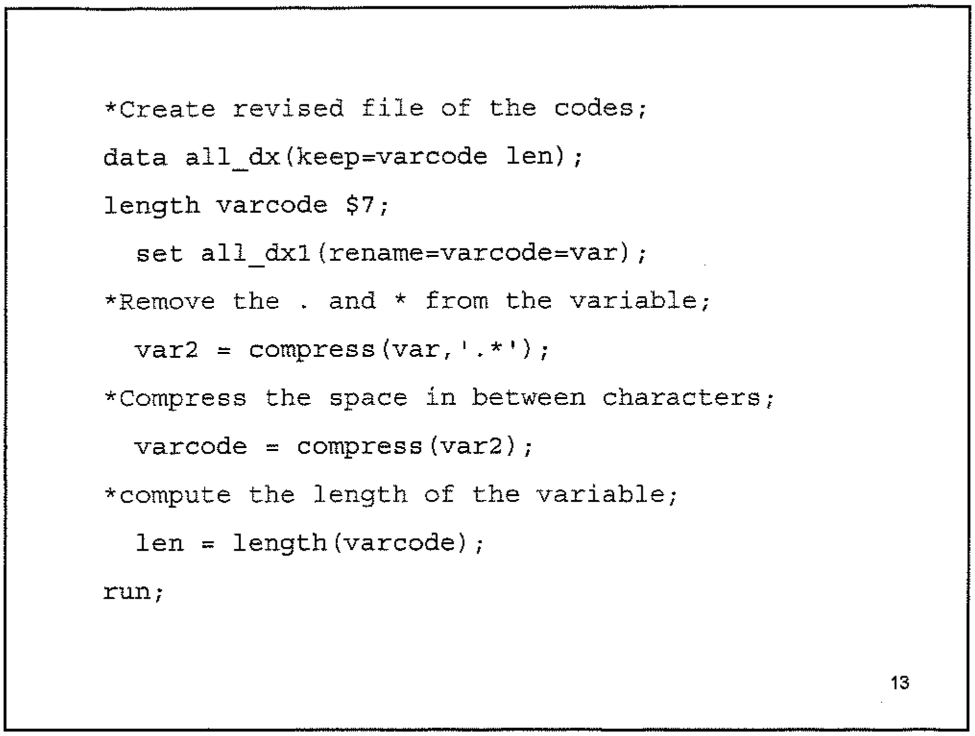


```
data icd9_war_toxic;
  input varcode$;
cards;
V58.2*
V58.61
246.3*
255.4*
285.1*
286.5*
286.7*
286.9*
307.81
336.1*
344*
```

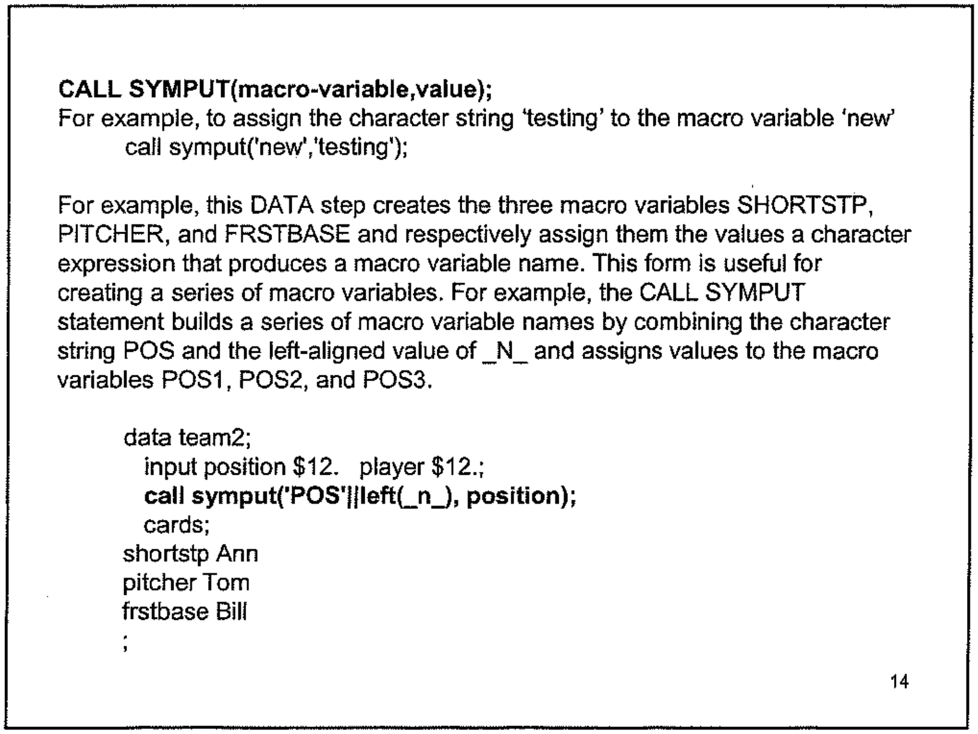
Create a text file of the codes

We have to remove the "." to do a match because our ICD-9 file does not have the decimal, it is implied after the first three characters

varcode
5693
5693
5997
42731
42731
V5861
2768



```
*Create revised file of the codes;
data all_dx(keep=varcode len);
length varcode $7;
  set all_dx1(rename=varcode=var);
*Remove the . and * from the variable;
  var2 = compress(var,'.*');
*Compress the space in between characters;
  varcode = compress(var2);
*compute the length of the variable;
  len = length(varcode);
run;
```

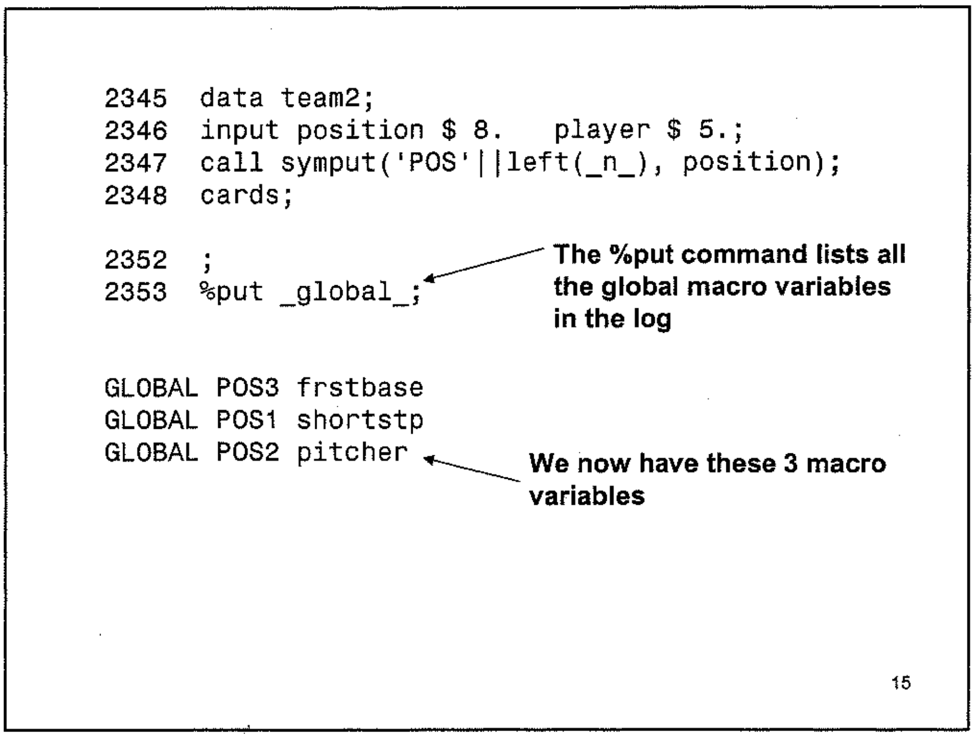


CALL SYMPUT(macro-variable,value);
For example, to assign the character string 'testing' to the macro variable 'new'

```
    call symput('new','testing');
```

For example, this DATA step creates the three macro variables SHORTSTP, PITCHER, and FRSTBASE and respectively assign them the values a character expression that produces a macro variable name. This form is useful for creating a series of macro variables. For example, the CALL SYMPUT statement builds a series of macro variable names by combining the character string POS and the left-aligned value of _N_ and assigns values to the macro variables POS1, POS2, and POS3.

```
data team2;
  input position $12.   player $12.;
  call symput('POS'||left(_n_), position);
  cards;
shortstp Ann
pitcher Tom
frstbase Bill
;
```



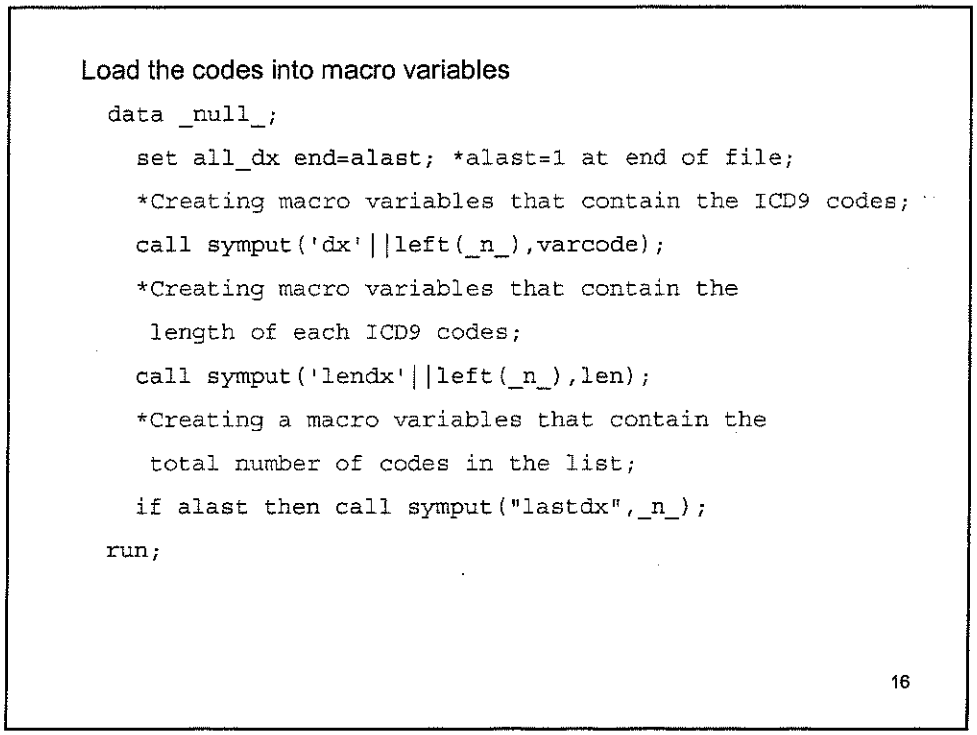
```
2345  data team2;
2346  input position $ 8.   player $ 5.;
2347  call symput('POS'||left(_n_), position);
2348  cards;

2352  ;
2353  %put _global_;
```

The %put command lists all the global macro variables in the log

```
GLOBAL POS3 frstbase
GLOBAL POS1 shortstp
GLOBAL POS2 pitcher
```

We now have these 3 macro variables

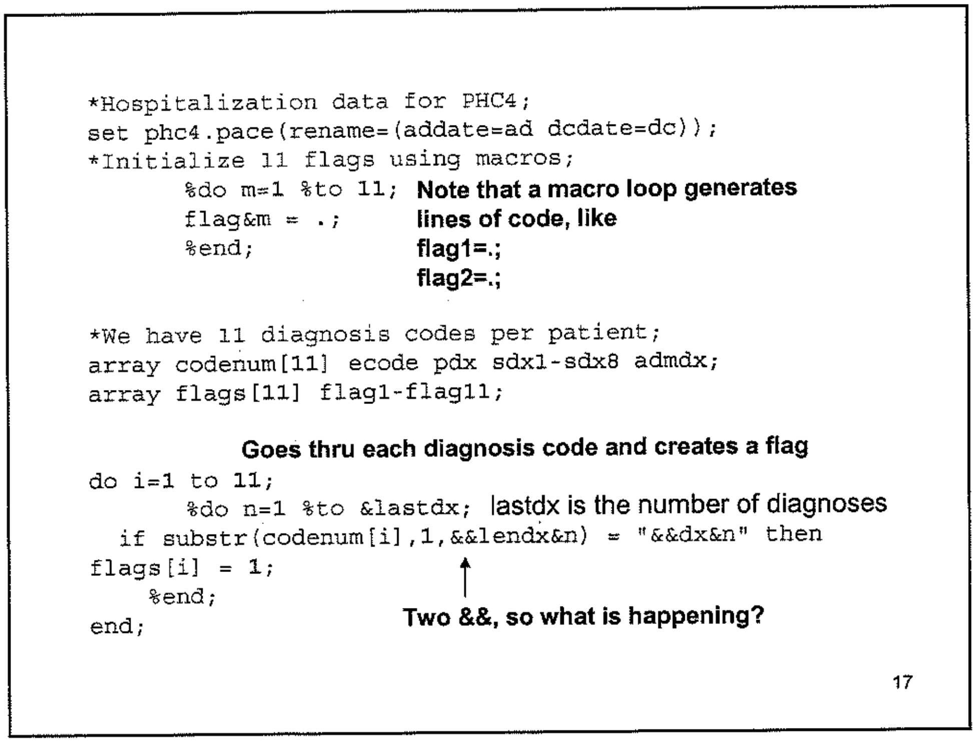
15



Load the codes into macro variables

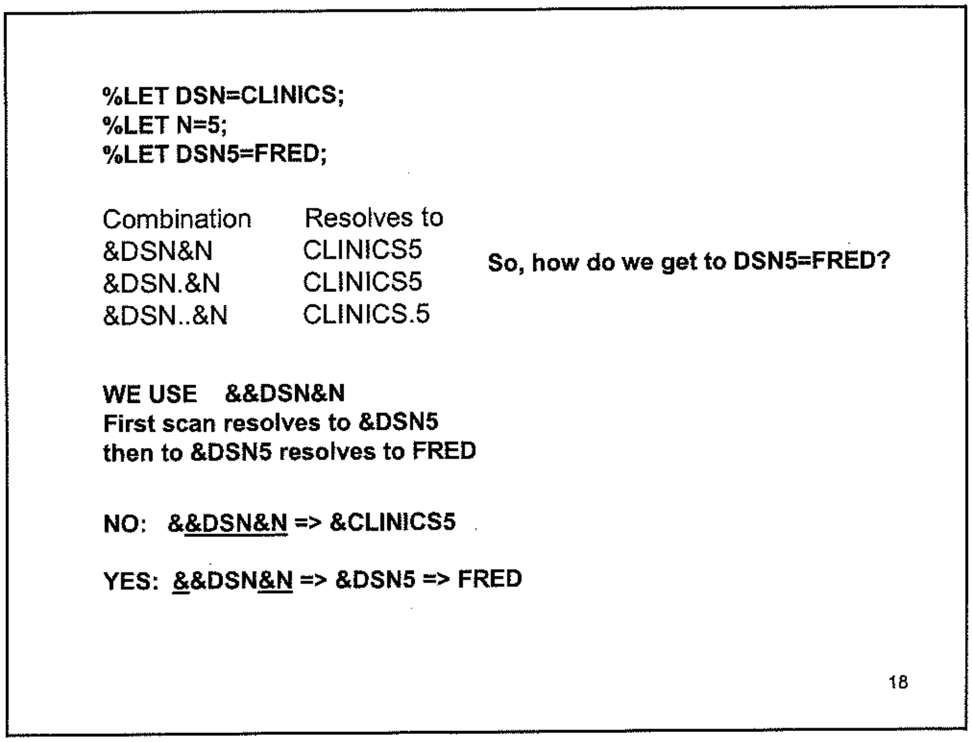
```
data _null_;
  set all_dx end=alast; *alast=1 at end of file;
  *Creating macro variables that contain the ICD9 codes;
  call symput('dx'||left(_n_),varcode);
  *Creating macro variables that contain the
   length of each ICD9 codes;
  call symput('lendx'||left(_n_),len);
  *Creating a macro variables that contain the
   total number of codes in the list;
  if alast then call symput("lastdx",_n_);
run;
```

16



```
*Hospitalization data for PHC4;
set phc4.pace(rename=(addate=ad dcdate=dc));
*Initialize 11 flags using macros;
      %do m=1 %to 11;  Note that a macro loop generates
      flag&m = .;      lines of code, like
      %end;            flag1=.;
                       flag2=.;

*We have 11 diagnosis codes per patient;
array codenum[11] ecode pdx sdx1-sdx8 admdx;
array flags[11] flag1-flag11;

          Goes thru each diagnosis code and creates a flag
do i=1 to 11;
      %do n=1 %to &lastdx;  lastdx is the number of diagnoses
  if substr(codenum[i],1,&&lendx&n) = "&&dx&n" then
flags[i] = 1;                       ↑
    %end;
end;                    Two &&, so what is happening?
```

17



%LET DSN=CLINICS;
%LET N=5;
%LET DSN5=FRED;

Combination	Resolves to
&DSN&N	CLINICS5
&DSN.&N	CLINICS5
&DSN..&N	CLINICS.5

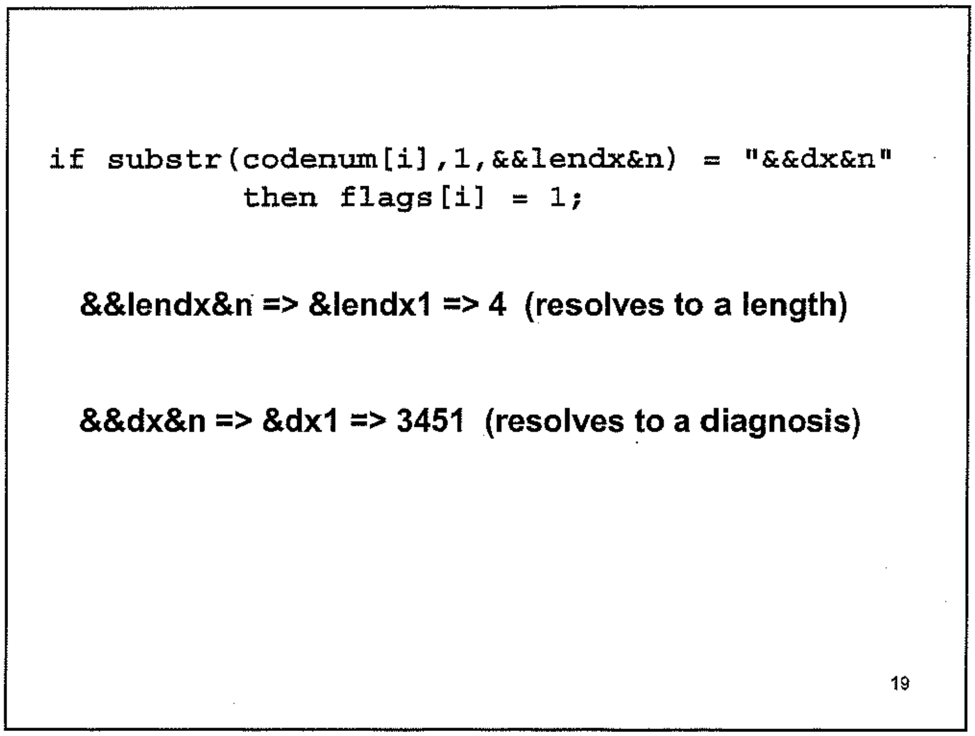
So, how do we get to DSN5=FRED?

WE USE &&DSN&N
First scan resolves to &DSN5
then to &DSN5 resolves to FRED

NO: &&DSN&N => &CLINICS5

YES: &&DSN&N => &DSN5 => FRED

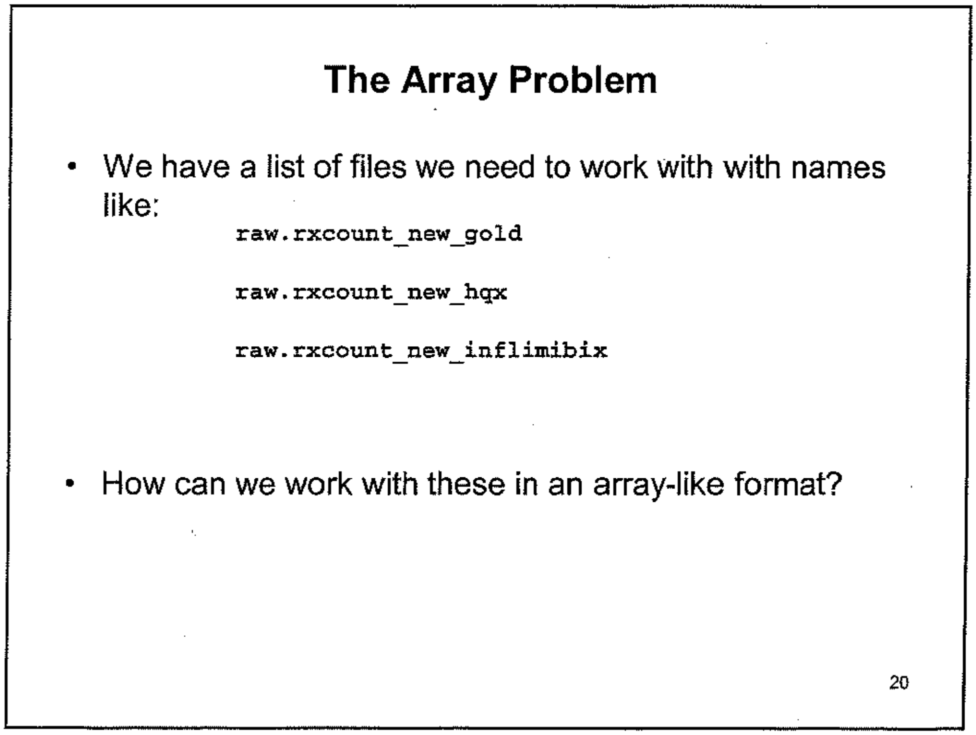
18



```
if substr(codenum[i],1,&&lendx&n) = "&&dx&n"
          then flags[i] = 1;
```

&&lendx&n => &lendx1 => 4 (resolves to a length)

&&dx&n => &dx1 => 3451 (resolves to a diagnosis)

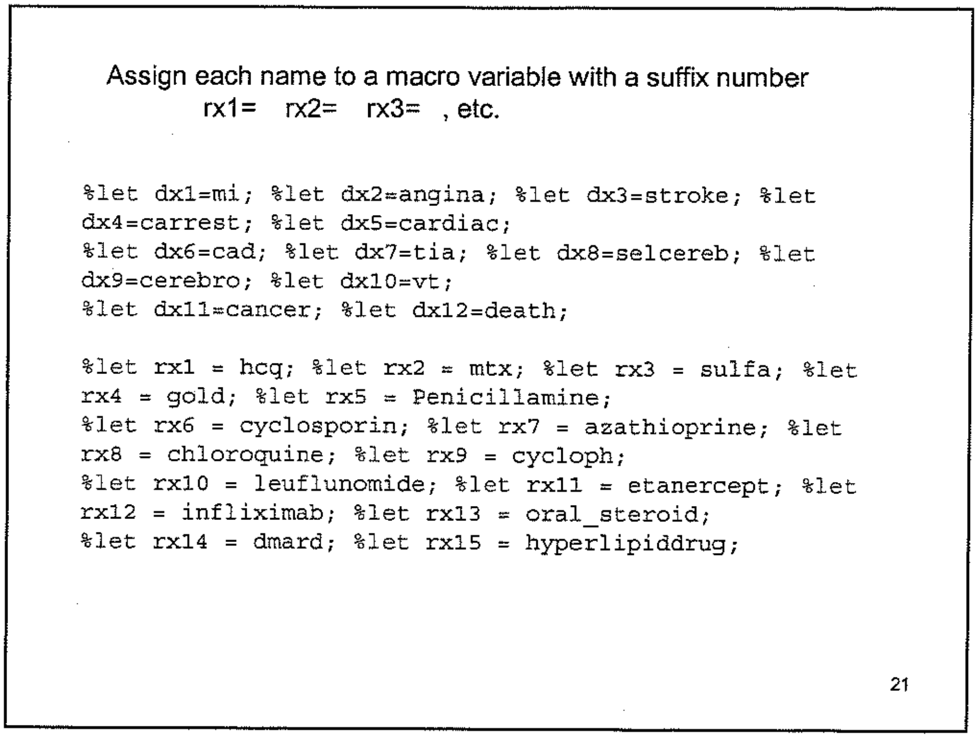


The Array Problem

- We have a list of files we need to work with with names like:

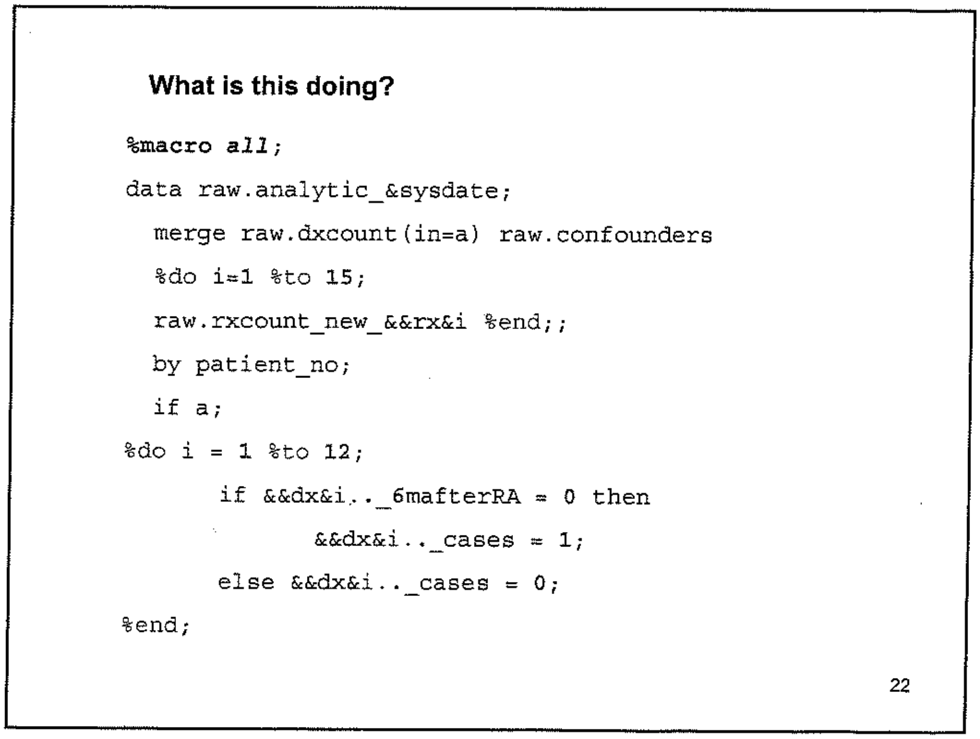
```
raw.rxcount_new_gold

raw.rxcount_new_hqx

raw.rxcount_new_inflimibix
```

- How can we work with these in an array-like format?



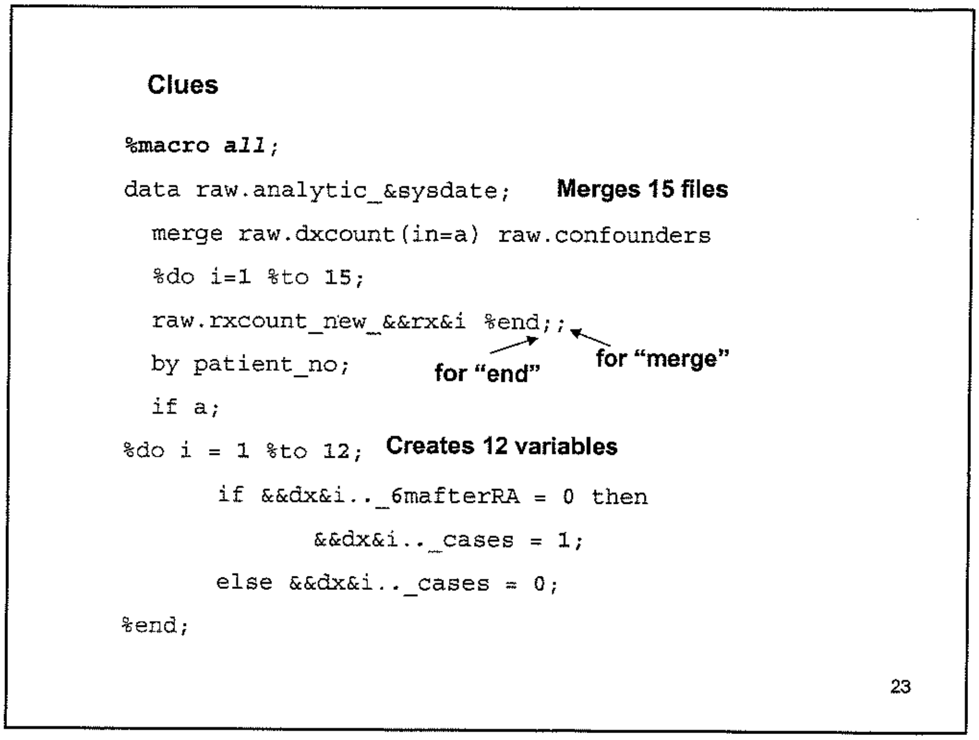
Assign each name to a macro variable with a suffix number
rx1= rx2= rx3= , etc.

```
%let dx1=mi; %let dx2=angina; %let dx3=stroke; %let
dx4=carrest; %let dx5=cardiac;
%let dx6=cad; %let dx7=tia; %let dx8=selcereb; %let
dx9=cerebro; %let dx10=vt;
%let dx11=cancer; %let dx12=death;

%let rx1 = hcq; %let rx2 = mtx; %let rx3 = sulfa; %let
rx4 = gold; %let rx5 = Penicillamine;
%let rx6 = cyclosporin; %let rx7 = azathioprine; %let
rx8 = chloroquine; %let rx9 = cycloph;
%let rx10 = leuflunomide; %let rx11 = etanercept; %let
rx12 = infliximab; %let rx13 = oral_steroid;
%let rx14 = dmard; %let rx15 = hyperlipiddrug;
```



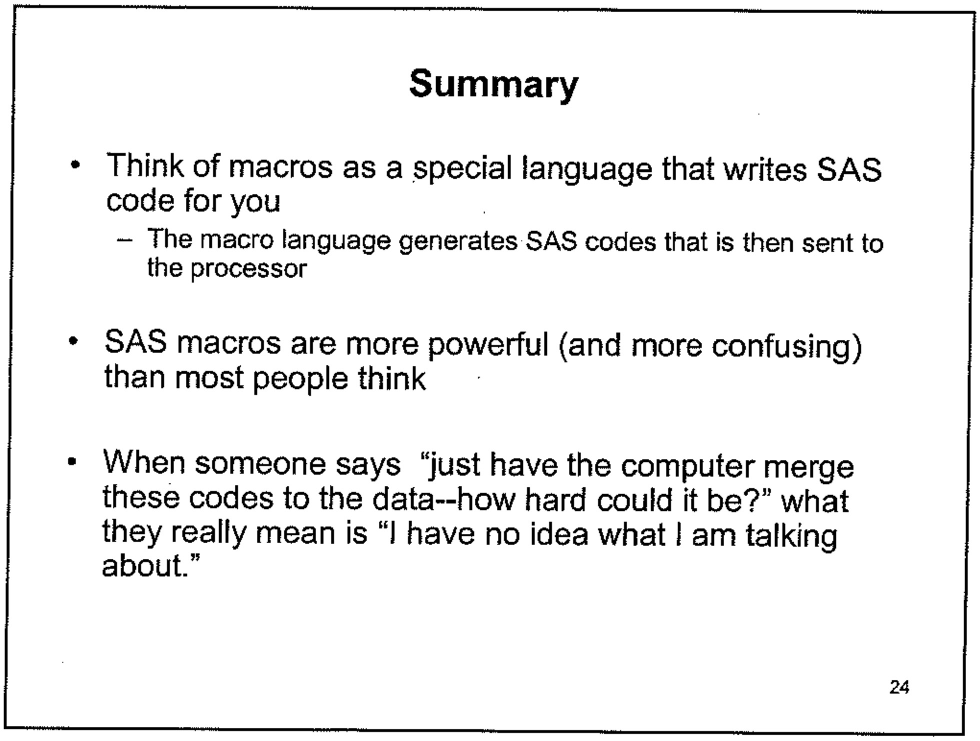
What is this doing?

```
%macro all;
data raw.analytic_&sysdate;
  merge raw.dxcount(in=a) raw.confounders
  %do i=1 %to 15;
  raw.rxcount_new_&&rx&i %end;;
  by patient_no;
  if a;
%do i = 1 %to 12;
      if &&dx&i.._6mafterRA = 0 then
             &&dx&i.._cases = 1;
      else &&dx&i.._cases = 0;
%end;
```



Clues

```
%macro all;
data raw.analytic_&sysdate;     Merges 15 files
  merge raw.dxcount(in=a) raw.confounders
  %do i=1 %to 15;
  raw.rxcount_new_&&rx&i %end;;
  by patient_no;          for "end"     for "merge"
  if a;
%do i = 1 %to 12;  Creates 12 variables
      if &&dx&i.._6mafterRA = 0 then
             &&dx&i.._cases = 1;
      else &&dx&i.._cases = 0;
%end;
```



Summary

- Think of macros as a special language that writes SAS code for you
  - The macro language generates SAS codes that is then sent to the processor
- SAS macros are more powerful (and more confusing) than most people think
- When someone says "just have the computer merge these codes to the data--how hard could it be?" what they really mean is "I have no idea what I am talking about."