Randomization, Logical Statements, and Proc TRANSPOSE

Example 1:
Random assignment of subjects to either a TREATMENT or CONTROL group

- ✓ **RANDOMIZATION is a process in which subjects (or patients) are allocated randomly to different treatments in a study.**
- ✓ **Randomized clinical trial is the epitome of all research designs because it provides the strongest evidence for concluding causations and the best insurance that the results were due to the intervention**

In any experiment, one of the most important steps is to randomly assign the subjects to either a treatment or control group to avoid the bias in the allocation of subjects to either of these two groups.

**********;

EXAMPLE17.SAS

This example randomly allocates 10 participants to either treatment or control group.
Note the use of
SEED = 0 in RANUNI statement (RANUNI uses the Uniform Distribution)
PROC RANK to create two groups
OUT =

```
*************************************************************************
This program is run once with a SEED = 0, & second TIME with a SEED = 34.

options nodate nonumber;

Proc format;
      Value groupf      0 = "CONTROL"
                        1 = "TREATMENT";
run;

data example17;
      input id name $25.;
      group = ranuni(0);      * Seed is 0. Therefore, every time the
                                program is run, we get a different
                                randomization;

datalines;
1 John, Smith
2 Amy, Burns
3 Kimberly, Clark
4 Eugene, Johns
5 Ana, Tran
6 Paula, Wolfgang
7 Rupert, Murdock
8 Issac, Newton
9 Jones, Patricia
10 George, Confused
;
```

Randomization, Logical Statements, and Proc TRANSPOSE

```
proc rank data= example17 groups = 2 out = new;
      var group;
run;

proc print data = new;
run;
```

Run up to this point and see what output you get.

It allocates equal number of participants to each group.

Suppose you add an 11th person, there will be unequal number of participants in groups.

```
proc sort data=new;
      by name;
run;

proc print data=new;
title 'Random allocation of subjects to TREATMENT or CONTROL
       group';
var id name group;
format group groupf.;
run;
```

Randomization, Logical Statements, and Proc TRANSPOSE

- **This following program uses SEED = 100,000 (any other number).**
- **Each time this program is run we get the same randomization.**
- **The only difference between this and the previous program is the value of the SEED**

```
options nodate nonumber;

Proc format;
      Value groupf      0 = "CONTROL"  1 = "TREATMENT";
run;

data temp;
      input id name $25.;
      group = ranuni(100000);

* Seed is NOT 0. Therefore, every time the program is run, we get the
  SAME randomization;

datalines;
1 John, Smith
2 Amy, Burns
3 Kimberly, Clark
4 Eugene, Johns
5 Ana, Tran
6 Paula, Wolfgang
7 Rupert, Murdock
8 Issac, Newton
9 Jones, Patricia
10 George, Confused
;
proc rank data=temp groups = 2 out = new;
      var group;
run;

proc sort data=new;
      by name;
run;

proc print data=new;
title 'Random allocation of subjects to TREATMENT or CONTROL
       group';
var id name group;
format group groupf.;
run;
```

Example 2: EXAMPLE17A.SAS

```
      data example17A;
            do patient = 1 to 100;
                  if ranuni (123456) le 0.5 then group = 'Drug A';
                        else group = 'Drug B';
                              output;
            end;
      run;
            proc print data=example17a;
                  title ' Assignment of patients randomly to two groups';
            run;
```

Example 3: Random Selection of Blocks

Sometimes we may have to select a certain number of people from different age groups such as 60 – 74 and 75 – 84 as in this example.

Study subjects can come from different STATES,
Within each state, different COUNTIES
Within each county, there may be different TRACTs
Within each TRACT, there may be BLOCKS
Within each block, there may be subjects belonging to Different age groups.

Following example limits to one state, one county and one tract, and within this track, the blocks were randomly selected. Then, we selected as many blocks as was needed to get the total number of subjects needed.

Note that we selected all the subjects in each age group who belonged to the selected blocks.

```
INPUT dataset contains STATE, COUNTY, TRACTNUM, BLOCKNUM, AGE60_74 (NUMBER OF
PEOPLE IN THE AGE GROUP 60 TO 74), AGE75_84 NUMBER OF PEOPLE IN THE AGE GROUP
75 TO 84), and TOTALNUM (TOTAL NUMBER IN BOTH AGE GROUPS)

Data were obtained from Census Bureau of the United States.

We need to select 20 blocks randomly and include in our study everybody in
each of these 20 blocks
```

The dataset, EXAMPLE18.DAT is in C:\DAT_FILE\;

EXAMPLE18.SAS

```
options nodate nonumber;
Data example18;

label       state = "State code"
            county = "County code"
            tractnum = "Tract number"
            blocknum = "Block number"
            age60_74 = "Number in 60 - 74 age range"
            age75_84 = "Number in 75 - 84 age range"
            totalnum = "Total number in each block";

      infile 'c:\dat_file\example18.dat';
```

Note the use of INFILE statement to refer to external data file

```
      input state county tractnum blocknum age60_74 age75_84 totalnum;

x=ranuni(0);
```

Randomization, Logical Statements, and Proc TRANSPOSE

Random numbers generated were stored in the variable X
These numbers fall between 0 and 1 since we have used the uniform distribution

```
*****************************************************************;
** In the SAS function, RANUNI(0), the argument 0 is called the seed.
   The value for the seed should be 0 in order to get a different
   series of random numbers each time the program is run.

   If we need the repeatable series of random numbers, then we need to
   give a 5- or 6- or 7-digit odd number as the value for the seed;
*****************************************************************;
y=x*100000;
```

Random numbers stored in X are all decimal numbers. In this statement we create a new variable Y that stores the values of X multiplied by 100000. This helps to look at the large numbers rather than decimal numbers. This statement is completely optional. Please run the program to print X as well as Y and see which one is easier to look at.

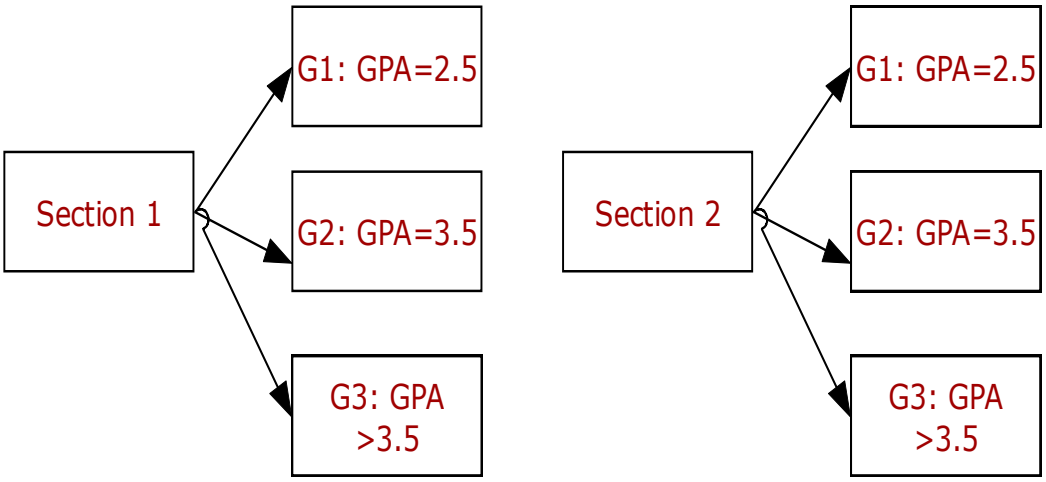
```
proc sort;
      by x;
run;

proc print data=example18 (obs=20); * This prints the first 20 records;

title ' Random Selection of Blocks from Tract 212';
var county tractnum blocknum age60_74 age75_84 x totalnum;
format y 7.;
run;
*****************************************************************;
```

Example 3: Randomization.

- **Please read carefully the description of the purpose of this program given in the flowchart.**
- **EXAMPLE19.SAS does accomplish the task.**
- **There is another most efficient way of doing the same task but using what is called "macros" in SAS which will be discussed later in the course if time permits.**



We need to select: 2 students from Group 1 (GPA = 2.5)
3 from Group 2 (GPA = 3.5)
5 from Group 3 (GPA > 3.5)

from each of the two sections

The following SAS program accomplishes this task:

EXAMPLE19.SAS

```
data example19;
input id section GPA;
datalines;
1 1 2.5
2 1 2.5
3 1 2.5
4 1 3.5
5 1 3.5
6 1 3.5
7 1 3.5
8 1 3.6
9 1 3.7
10 1 4.0
11 1 3.8
12 1 3.8
13 1 3.7
14 1 4.0
15 1 3.6
16 1 4.0
17 2 2.5
18 2 2.5
19 2 3.5
20 2 3.5
21 2 3.5
22 2 3.5
23 2 3.6
24 2 4.0
25 2 3.9
;
run;
```

The following three SAS statements create a data set called TEMPP by taking only those records from the original data set EXAMPLE19 that belong to Section 1 because of IF statement, IF SECTION = 1;

```
data tempp;
      set example19;
if section=1;
************************************************************;
```

Randomization, Logical Statements, and Proc TRANSPOSE

The following segment creates a SAS data set by name S1_GPA2_5 for SECTION 1.

S1_GPA2_5 contains the 2 randomly selected records for GPA = 2.5 from Section 1;

```
data temp1;
      set tempp;

if gpa=2.5;
      x = ranuni(0);
output;
```

TEMP1 contains only those records from TEMPP that have a value of 2.5 for GPA.

Each of these records is now assigned a random number stored in the variable X generated using RANUNI.

Note that the OUTPUT statement is responsible for retaining each record as it is read. To understand this, run the program without using the OUTPUT statement and again, using the OUTPUT statement to see the effect of OUTPUT;

```
proc sort data=temp1;
      by x;
run;

data S1_GPA2_5;
      set temp1 (obs=2);
run;
```

Data set S1_GPA2_5 contains only 2 records that were randomly selected from TEMP1. NOTE the use of OBS=2 in the SET statement.

Summary of what the program has done so far:

1. **Created TEMPP that has only those records that belong to Section 1.**
2. **TEMP1 contains only those records from TEMPP that have GPA=2.5 and each record in TEMP1 has been assigned a random number.**
3. **Date set, S1_GPA2_5 , has now 2 records that were randomly selected from TEMP1.**

Thus, we have the data set called S1_GPA2_5 that has 2 randomly selected records with GPA=2.5 from Section 1;

```
**************************************************************************;
```

Note:

We do exactly same as above for 2 other levels of GPA for SECTION 1. Then, repeat it for SECTION 2.

The following segment is for GPA = 3.5 and create a SAS data set called S1_GPA3_5.

```
data temp2;
      set tempp;
if gpa=3.5;
      x = ranuni(0);
      output;

proc sort data=temp2;
      by x;
run;

data S1_GPA_3_5;
      set temp2 (obs=3);
run;
```

Similarly, the above 11 SAS statements create a data set called S1_GPA3_5 that has 3 randomly selected records with GPA=3.5 from Section 1;

```
************************************************************;
```

The following segment is for GPA > 3.5 and create a SAS data set called S1_GPA_GT3_5

```
data temp3;
      set tempp;
if gpa gt 3.5;
      x = ranuni(0);
      output;

proc sort data=temp3;
      by x;
run;

data S1_GPA_GT3_5;
      set temp3 (obs=5);
run;
```

Similarly the above 11 SAS statements create a data set called S1_GPAGT3_5 that has 5 randomly selected records with GPA > 3.5 from Section 1;

```
************************************************************;
```

The following three SAS statements create a data set called FOR_SECTION1 that has 10 records (2 from S1_GPA2_5, 3 from S1_GPA3_5 and 5 from S1_GPA_GT3_5) for Section 1;

```
data FOR_SECTION1;
      set s1_gpa2_5 s1_gpa3_5 s1_gpa_gt3_5;
run;

*************************************************************;
```

Randomization, Logical Statements, and Proc TRANSPOSE

```
*************************************************************;
data tempp;
      set example19;
if section=2;            * Creates a dataset for SECTION 2;
*************************************************************;
      * The following segment is for GPA = 2.5;
data temp1;
      set tempp;
if gpa=2.5;
      x = ranuni(0);
      output;

proc sort data=temp1;
      by x;
run;

data s2_gpa2_5;
      set temp1 (obs=2);
run;
*******************************************;
* The following segment is for GPA = 3.5;
data temp2;
      set tempp;
if gpa=3.5;
      x = ranuni(0);
      output;

proc sort data=temp2;
      by x;
run;

data s2_gpa3_5;
      set temp2 (obs=3);
run;
*******************************************;
* The following segment is for GPA > 3.5;
data temp3;
      set tempp;
if gpa gt 3.5;
      x = ranuni(0);
      output;

proc sort data=temp3;
      by x;
run;

data s2_gpa_gt3_5;
      set temp3 (obs=5);
run;
*************************************************************;
data for_section2;
      * This creates one datafile for SECTION 2 with
        randomly chosen records for all 3 levels of of GPA;
      set s2_gpa2_5 s2_gpa3_5 s2_gpa_gt3_5;
run;
*************************************************************;
```

Randomization, Logical Statements, and Proc TRANSPOSE

The following SAS statements create a data set called FINAL.SAS7BDAT that has a total of 20 records (10 records for Section 1 and 10 records for Section 2).

```
data final;
      set for_section1 for_section2;
run;

*************************************************************;
proc print data=final;
      title 'Tabulation of selected records';
run;
*************************************************************;
```

Note:

The program that follows now in the next couple of pages do exactly same as EXAMPLE19.SAS above but using what is called MACROS which is a separate topic we will do at a later stage in the semester.

IF-THEN/ELSE and DO-END statements

IF – THEN – ELSE are logical statements. DO and END statements are used to repeatedly perform a certain task.

Please note that there should be as many IF as the number of ELSEs. This is one way checking that there is no infinite loop in the program.

Also note that IF – THEN – ELSE are used when there are mutually exclusive conditions. For example a person cannot belong to both MALE and FEMALE categories. Basically, a person or an entity cannot belong to two mutually exclusive groups.

> **Please read the description of the problem given below.**
>
> **Another important factor you need to note is to make sure that missing values are taken care of.**

EXAMPLE:

Suppose we have a variable *AGE* whose values range from 30 to 65 years. Also, suppose that we have data on *AGE* for 65 subjects and no data on *AGE* for 7 subjects (*Total sample size is 72*)

Suppose we like to create a new variable *AGEGROUP* that has 3 levels:

- Level 1 has subjects whose ages are **from 30 to 45 years**
- Level 2 has subjects whose ages fall **between 45 and 55**
- Level 3 has subjects whose ages are **greater than or equal to 55**

- **Note the mutually exclusive categories above**
- **Note, however, that AGEGROUP has seven missing values.**

The following SAS statement will create the new variable *AGEGROUP*:

```
if age = . then agegroup = .;
      else if (30 ≤ age ≤ 45) then agegroup = 1;
            else if (45 < age < 55) then agegroup = 2;
                  else agegroup = 3;
```

```
Then, we need the following "value" and "format" statements
that should go at appropriate places.
```

```
      value agegrpf     1 = "Group 1: 30 ≤ AGE ≤ 45"
                        2 = "Group 2: 45 < AGE < 55"
                        3 = "Group 3: AGE ≥ 55 years";

      format agegroup agegrpf.;
```

Randomization, Logical Statements, and Proc TRANSPOSE

The conditions in the ***IF-THEN/ELSE*** statement can be one of the following comparisons:

Operator	**Meaning**
EQ (=)	Equal to
NE (~=)	Not equal to
LT (<)	Less than
GT (>)	Greater than
GE (≥)	Greater than or equal to
LE (≤)	Less than or equal to

In addition, the comparisons can be combined into a more complex condition using Boolean operators *AND*, *OR*, and *NOT*.

In the previous example we created a SINGLE categorical variable using IF – THEN – ELSE statements but did not use any DO …END statements.

The next example illustrates the use of DO ---- END statements in addition to IF – THEN – ELSE statements.

SAS program in which IF-THE/ELSE, WHERE, SUBSETTING IF, and DO-END statements are used

This is an example in which we create two categorical variables (DUMMY variables) for each condition. That is the reason we need DO - - - -END statements.

EXAMPLE20.SAS

```
options ps=55 ls=64 nodate nonumber;

proc format;
      value yesnof       1 = "Yes"
                         0 = "No";

      value exptypef     1 = "Not Exposed"
                         2 = "Previously exposed"
                         3 = "Currently exposed";
*****************************************************************
      An example of the use of IF-THEN/ELSE and DO-END statements
      to create two DUMMY variables for a variable that has 3 levels
*****************************************************************;
data example20;

input id exp_type @@;    * Note the use of @@ symbol;

label
      exp_type = "Type of Exposure"
      prev_exp = "Previously exposed?"
      cur_exp = "Curently exposed?";
```

EXP_TYPE is the ORIGINAL variable that has 3 levels.

We want to create two dummy variables (named PREV_EXP and CUR_EXP) based on the values of EXP_TYPE

(PREV_EXP = Previously exposed and CUR_EXP = Currently exposed)

CUR_EXP = 1	**if a subject has EXP_TYPE = 3, otherwise CUR_EXP = 0**
PREV_EXP = 1	**if a subject has EXP_TYPE = 2, otherwise PREV_EXP = 0**
CUR_EXP = 0 and PREV_EXP = 0	**If a subject has EXP_TYPE = 1**
CUR_EXP = . and and PREV_EXP = .	**if a subject has EXP_TYPE = .**

```
format      exp_type exptypef. prev_exp cur_exp yesnof.;

if exp_type =. then do;
                     cur_exp=.;
                     prev_exp=.;
                    end;
   else if exp_type = 2 then do;
                              cur_exp=0;
                              prev_exp=1;
                             end;
           else if exp_type = 3 then do;
                                      cur_exp=1;
                                      prev_exp=0;
                                     end;
                  else do;
                        cur_exp=0;
                        prev_exp=0;
                       end;
datalines;
1 1 2 1 3 1 4 1 5 1 6 . 7 1 8 1 9 1 10 1 11 2 12 2 13 2 14 . 15 2 16 2 17 .
18 2 19 2 20 2 21 3 22 . 23 . 24 3 25 3 26 3 27 3 28 3 29 3 30 3
;
*************************************************************;
proc print data=example20;
      title 'Tabulation of the data from EXAMPLE20 data set';
run;

proc freq data=example20;
      title 'Frequency Tabulation from EXAMPLE20 data set';
      tables exp_type prev_exp cur_exp;
run;

proc freq data=example20;
title 'Frequency Tabulation from EXAMPLE20 based on WHERE
 statement';
      tables exp_type;
      where (exp_type=2 or exp_type=3);
run;
*************************************************************;
data dataset1;

      set example20;
      if (exp_type = 2 or exp_type= 3);

proc freq data= dataset1;
      title 'Frequency Tabulation from DATASET1 data set';
      title2 'created based on SUBSETTING IF statement';
      tables exp_type;
run;
```

WHERE Statement

In the above program, note the use of *WHERE* statement. The *WHERE* statement selects the observations that satisfy the logical condition placed after the keyword *WHERE*. In the above example, the last **P**roc **F**req is performed only on those observations for which *EXP_TYPE* is either **2** or **3**.

Subsetting IF Statement

In the last part of the above program, we have used ***IF*** statement to create a new SAS data set. The *IF* statement that is used here is called a *subsetting IF* statement.

The *SET* statement is used to create a new SAS data set called *DATASET1* from the original SAS data set *EXAMP16*. This new data set *DATASET1* contains only those observations for which *EXP_TYPE* is either **2** or **3**.

Two DUMMY variables created in the above program

We started out with ***EXP_TYPE*** variable taking values **1** (*not exposed*), **2** (*previously exposed*), and **3** (*currently exposed*).

- The dummy variable ***PREV_EXP*** takes on a value of **1** (*Yes, previously exposed*) when ***EXP_TYPE*** = ***2***, otherwise, ***PREV_EXP*** = **0** (*No, not previously exposed*).

- The dummy variable ***CUR_EXP*** takes on a value of **1** (*Yes, currently exposed*) when ***EXP_TYPE*** = ***3***, otherwise, ***CUR_EXP*** = **0** (*No, not currently exposed*).

- Both dummy variables ***PREV_EXP*** and ***CUR_EXP*** take a value of **0** when ***EXP_TYPE*** = ***1***.

- Both dummy variables ***PREV_EXP*** and ***CUR_EXP*** take a value of **.** (*period*) when ***EXP_TYPE*** = **.** (that is, a *missing value*).

```
************************************************************;
```

DO UNTIL- - - -END statement

The purpose of this example is to compute the following for each person. This is for ID = 1

		Difference between two dates
2/3/1998	2/3/2000	730.00
2/3/2000	2/3/2002	731.00
2/3/2002	2/3/2007	1826.00
	Average	1095.67

```
* Use of DO UNTIL ....END statement to find the average number
  of days between visits;
```

EXAMPLE21.SAS;

```
************************************************************;
data example21;
      input id visit date mmddyy10.;
      format date mmddyy10.;
      datalines;
1 1 02/03/1998
1 2 02/03/2000
1 3 02/03/2002
1 4 02/03/2007
2 1 02/03/1996
2 2 02/03/1998
2 3 02/03/2000
2 4 02/03/2005
3 1 02/03/1991
3 2 02/03/1995
3 3 02/03/1999
3 4 02/03/2001
4 1 02/03/1989
4 2 02/03/1993
4 3 02/03/1995
4 4 02/03/1998
;
```

* **Note that for each person there are 4 visit dates.**

  For each person, we want to compute the <u>number of days</u>
  between Visit 1 Date and Visit 2 Date
  between Visit 2 Date and Visit 3 Date
  between Visit 3 Date and Visit 4 Date

  Then, take the average of these numbers.

  I have shown the computation for one person above;

```
run;
```

Randomization, Logical Statements, and Proc TRANSPOSE

```
data result;
      n = 0;
```

The above line creates a variable n whose initial value is 0.

```
do until (last.id);
      set example21;
      by id date;
      if first.id then firstday = date;
```

This statement takes the first date for each ID and puts it in the variable called FIRSTDAY;

```
      n + 1;
end;
```

The 4 statements that are between DO UNTIL (LAST.ID) and END are carried out for each person until the last record for each person is processed.

```
if n > 1 then avgdays = (date - firstday)/(n - 1);
drop n visit;
*************************************;
proc print data=result;
      title 'Tabulation of the results';
      format date firstday mmddyy10.;
run;
*************************************;
```

Use of PROC TRANSPOSE instead of DO UNTIL - - - - END Statements

EXAMPLE21A.SAS

* Following program does exactly same as what was done by EXAMPLE21.SAS except that this one uses PROC TRANSPOSE

```
*************************************************************;
* Use of PROC TRANSPOSE procedure to find the average number
  of days between the first and the last visit for each patient;
*************************************************************;
data example21A;
      input id visit date mmddyy10.;
      format date mmddyy10.;
datalines;
1 1 02/03/1998
1 2 02/03/2000
1 3 02/03/2002
1 4 02/03/2007
2 1 02/03/1996
2 2 02/03/1998
2 3 02/03/2000
2 4 02/03/2005
3 1 02/03/1991
3 2 02/03/1995
3 3 02/03/1999
3 4 02/03/2001
4 1 02/03/1989
4 2 02/03/1993
4 3 02/03/1995
4 4 02/03/1998
;
run;
*********************************************;
data one;
      set example21a (keep = id date);
run;
```

* The above three statements create a data set called ONE using the data for the variables ID and DATE from EXAMPLE21A;

```
proc sort data=one;
      by id;
run;
```

* The above three statements sort the data set ONE by ID;

Randomization, Logical Statements, and Proc TRANSPOSE

```
proc transpose data=one out=two prefix = DATE_VISIT;
      by id;
run;
```

* **The above three statements create a data set called TWO using the data set ONE by transposing all four dates for each person from ONE and putting them in 4 new columns (whose names are DATE_VISIT1, DATE_VISIT2, DATE_VISIT3 and DATE_VISIT4 because of the statement** prefix = DATE_VISIT) **and thus, creates one record per person;**

```
data final;
      set two(drop = _name_);
```

* **The above two statements create a data file called FINAL from the data set, TWO created above in the PROC TRANSPOSE;**

```
avg_num_days = (date_visit4 - date_visit1)/3;
```

* **The above statement computes the number of days between the first date and the last date for each person and divides that number by 3 to get the average.;**

```
run;

proc print data=final;
run;
```