PharmaSUG 2012 - Paper AD06

A SAS® Tool to Allocate and Randomize Samples to Illumina Microarray Chips

Huanying Qin, Baylor Institute of Immunology Research, Dallas, TX

Greg Stanek, STEEEP Analytics, Baylor Health Care System, Dallas, TX

Derek Blankenship, Quantitative Sciences, Baylor Health Care System, Dallas, TX

ABSTRACT

DNA/RNA Microarray has become a common tool for identifying differentially expressed genes under different experimental conditions. Several microarray platforms exist and differ by probe implementation and target-labeling strategies. When it comes to sample allocation, Illumina BeadChip is most unique because each chip can hold 12 samples, while for other platforms, each chip holds one sample or a combination of 2 paired samples. This adds complexity to sample randomization because each chip serves as a block and often the number of samples for each study group is not balanced, ie. each group does not have the same number of samples. The SAS PROC PLAN procedure is able to aid in the design and randomization of sample allocation for balanced studies but is limited for unbalanced studies. This paper describes a SAS macro tool we developed to implement an optimizing algorithm to allocate and randomize samples to Illumina chips for unbalanced study designs.

INTRODUCTION

It is important to randomly assign samples to different treatment groups in order to reduce the likelihood of bias due to factors other than treatment. SAS PROC PLAN or PROC SURVEYSELECT can be used for balanced design or when a fixed allocation ratio exists. To randomly allocate samples of different groups to fill in spaces has similar implication as assigning samples to treatment groups. However, the filling in space problem can be more complicated in real situation because often the sample size in each group is not the same and there is no fixed allocation ratio, especially when the block factor is involved.

Microarray experiments involve loading samples to microarray chips where DNA probes are embedded. These chips then go through hybridization step followed by a stream of work. Most microarray platforms use one chip for one sample or a mixture of two samples. However, Illumina BeadChip Human V4 is very unique in its capacity to hold 12 samples on one chip, which has great advantage in cost-effectiveness. But this also presents great challenge in sample allocation, because each chip serves as a block. We try to avoid scenarios where samples from the same study group are all allocated to same chips and want to maximize the number of study groups represented on each chip. For most studies the sample size of each group is not divisible by the number of chips needed. In the past, our collaborators used EXCEL to manually randomize the samples by trial and error until a desired sample allocation was obtained. The process can be very time consuming without consistently applying an algorithm. The SAS tool in this paper uses the macro language and PROC SQL to implement a fixed algorithm to optimize the randomization. This greatly simplifies the sample allocation problem for researchers.

PROGRAM FLOW

Two major steps are involved in the use of the tool:

1. Calculate total number of chips needed and fill in all spaces. This requires determining the number of chips for each study group and the number of samples from each group for each chip.
2. Randomize chips and sample locations within each chip.

1. Calculating Total Number of Chips

We developed an algorithm using linear programming to solve for 5 parameters:

1. Total number of chips needed.
2. Chips and each sample group need to be divided into 2 subgroups so that one subgroup of samples can be evenly assigned to the first group of chips and the other subgroup can be evenly assigned to the remaining chip group.

$$Chip_1 Size_1 + Chip_2 Size_2 = Sample\ Count$$
$$Size_1 - Size_2 = 1$$
$$Chip_1 + Chip_2 = Chip\ Count$$

$$where:\ Size_1 = Ceiling\left(\frac{Sample\ Count}{Chip\ Count}\right)$$

With some algebra we obtain the following equations that are integrated in the code to capture the unknown values:

$$Chip_1 = Sample\ Count + Chip\ Count(1 - Size_1),$$
$$Size_2 = Size_1 - 1\ and$$
$$Chip_2 = Chip\ Count - Chip_1$$

As an example, Table 1 contains a microarray study data that will process 185 samples for 4 groups, which requires 16 Illumina chips (ceiling[185/12]) since each chip holds 12 samples. Table 2 contains the calculated chips for each site necessary based upon the algorithm for the assignment and randomization within each chip.

Group	Sample Site	Sample Count
1	Seattle control	39
2	Seattle patient	38
3	Texas control	44
4	Texas patient	64

Table 1. Study example before calculating chips

Group	Sample Site	Sample Count	Chip Count	Sample Count/ Chip Count	Chip1	Size1	Chip2	Size2
1	Seattle control	39	16	2.4375	7	3	9	2
2	Seattle patient	38	16	2.375	6	3	10	2
3	Texas control	44	16	2.75	12	3	4	2
4	Texas patient	64	16	4	16	4	0	0
5	NULL	7	16	0.4375	7	1	9	0

Table 2. Table sample2 after calculating chips

Table 2 illustrates the 4th group is an ideal since it returns an integer value i.e., its size 64 is divisible by the number of chips (16) and we can easily assign 4 samples from this group to each chip. The other 3 groups do not return an integer value and their sizes vary as well, which requires the additional group 5 of 7 null spaces to be defined to balance out the allocation of samples on the chips for randomization. Given the samples from group 2 to 5 can't be evenly assigned to each chip, we need to determine how to divide those samples so that they will be randomly allocated to those chip spaces as much as possible.

The code is relatively simple where it uses Proc SQL to generate macro variables to be passed thru to the algorithm to calculate the chip and size variables described in Table 2.

It starts out by reading in the the data from Excel and restricting to the Site and generating the count and Group variables as shown in Table 1.

```
libname Illumina "\\directory\Path\Samples for randomization.xls" stringdates=yes;

 proc sql;
  create table sample as
  select  status as Site  label='Sample Site',
          count(*)  as Sample Label='Sample Count'
  from Illumina."Sheet1$"n (keep=Status)
  group by 1;
  quit;

libname Illumina clear;

 data sample;
  set sample;
  Label GRPS='Group';
  GRPS=_n_;
 run;
```

The first portion of the Macro (%Sample_Chips) utilizes Proc SQL to generate macro variables to calculate the size and chip variables in the next step. In the macro variable development we use the calculated function within SQL to sequentially build variables, the ceil function to capture the largest value for the chips and then assign them to macro variables.

The second portion of Macro Sample_Chips generates the variables for size and chips as described previously. The second portion calculates the size and chips for both an unbalanced sample (our example) and a balanced sample.

```
%macro sample_chips;
%global init_samp k chip tot_samp samp_diff;

  proc sql noprint;
  select sum(sample)                                as init_samp,
         ceil(calculated init_samp/12)              as chip,
         calculated chip*12                         as tot_samp,
         calculated tot_samp-calculated init_samp   as samp_diff,
         strip(put(COUNT(DISTINCT grps),8.))        as init_grps

  into: init_samp, :chip,  :tot_samp,   :samp_diff, :k
  from sample;
  quit;

%put &init_samp &chip &tot_samp &samp_diff &k;

%if %eval(&samp_diff) >0 and %eval(&samp_diff) < 12 %then %do;

  data tmp;
   x=1;
  run;

  proc sql;
  create table sample2 as
  select grps as Group, Site,
        Sample label='Sample Count',
        &chip as Chip,
        sample/&chip as grp_ratio  label='Sample to Chip Ratio',
        &chip*(1-ceil(sample/&chip))+sample as Chip1,
        ceil(sample/&chip) as Size1,
        &chip-calculated chip1  as Chip2,
        case when calculated chip1<&chip then ceil(sample/&chip)-1
        else 0 end as  Size2
  from sample
  union
  select &k+1 as grps as Group, 'NULL' as Site,
        &samp_diff as Sample label='Sample Count',
        &chip as Chip,
        &samp_diff/&chip as grp_ratio  label='Sample to Chip Ratio',
        &chip*(1-ceil(&samp_diff/&chip))+&samp_diff as Chip1,
        ceil(&samp_diff/&chip) as Size1,
        &chip-calculated chip1  as Chip2,
        case when calculated chip1<&chip then ceil(&samp_diff/&chip)-1
        else 0 end as Size2
  from tmp;
  quit;

  proc datasets library=work nolist; delete tmp; quit;
  %end;
```

Although it might be rare, there are situations where the samples are all balanced and therefore the following code will deal with the balanced case.

```
%else %if %eval(&samp_diff)=0 %then %do;
  proc sql;
  create table sample2 as
  select grps as Group, Site, sample label='Sample Count',
```

```
        &chip as Chip,
        sample/&chip as grp_ratio  label='Sample to Chip Ratio',
        &chip*(1-ceil(sample/&chip))+sample as Chip1,
        ceil(samp/&chip) as Size1,
        &chip-calculated chip1  as Chip2,
        case when calculated chip1<&chip then ceil(sample/&chip)-1
        else 0 end as Size2
  from sample;
  quit;
%end;
%mend sample_chips;
%sample_chips;
```

The table sample2 (See Table 2) is the key for sample allocation and provides detailed information on how to divide the chips and samples. As an example, for the first group, each of the first 7 chips should have 3 samples and the rest of 9 chips should have 2 samples on each.

However, guided by these parameters, there is multiple ways to allocate all samples to fill in the spaces of all chips. Further work is needed to automate the process for selecting the best way. In this paper, we randomly picked one combination (Table 3) to implement and a simple Proc SQL step was used to create a dataset according to Table 3.

	Number of samples by group					
Chip	Group1	Group2	Group3	Group4	Group5	Total
1	4	3	2	2	1	12
2	4	3	2	2	1	12
3	4	2	2	3	1	12
4	4	2	2	3	1	12
5	4	2	3	2	1	12
6	4	2	3	2	1	12
7	4	2	3	2	1	12
8	4	2	3	3	0	12
9	4	2	3	3	0	12
10	4	2	3	3	0	12
11	4	2	3	3	0	12
12	4	2	3	3	0	12
13	4	3	3	2	0	12
14	4	3	3	2	0	12
15	4	3	3	2	0	12
16	4	3	3	2	0	12
Total	64	38	44	39	7	192

Table 3. The blueprint for initial allocation of samples to fill in the chips

```
data assignment(drop=i j);
 do i=1 to &chip.;
 chip=i;
 do j=1 to 12;
 location=j; output;
end;
end;
run;

proc sql;
create table assignment as
select chip, location,
```

```
case when location <=4    and 1<=chip<=16 then 4
when location=5           and 1<=chip<=7 then 5
when location=5           and 8<=chip<=16 then 3
when location in (6, 7)   and 1<=chip<=16 then 3
when location=8           and 5<=chip<=7 then 3
when location=8           and (1<=chip<=4 or 8<=chip<=16) then 2
when location=9           and 1<=chip<=16 then 2
when location=10          and chip in (1,2,5,6,7,13,14,15,16) then 2
when location=10          and chip in (3,4,8,9,10,11,12) then 1
when 10<location          and 1<=chip<=16 then 1 end as GRP
from assignment
order by chip;
quit;
```

2. Randomize Chips And Sample Locations Within Each Chip

After the first step, all samples of the same group will be clustered with each other, and the next step will be to randomize the chip ID and sample location within each chip. We use random number generator functions to accomplish this purpose.

In the codes below, the data set step1 comes from merging between the dataset assignment and the original sample list from the researcher and it has variables for sample ID, sample group status, chip ID and chip location according to information in Table 3. First, samples are randomized within each chip using the rand function followed by a few data manipulation steps.

```
proc sort data=step1; by chip location grp; run;

data step2;
 set step1;
 random_num=rand('uniform');
run;

proc sort data=step2;  by chip random_num; run;

proc rank data=step2 out=step3(drop=location rename=(location_new=location));
 by chip;
 var random_num;
 ranks location_new; label location =' ';
run;

data step4;
 do i=1 to &chip.;
 chip=i;
 random_chip=rand('uniform'); output;
 end;
run;

proc sql;
create table step5 as
select  a.*,  b.random_chip
from step3 a left  join step4 b
on a.chip=b.chip
order by random_chip;
quit;
```

It may not be required to randomize the chips every time, but it is a good practice especially when chips from different batches are used for one experiment.

```
data step6 (drop=chip rename=(new_chip=chip));
 set step5;
 by random_chip;
 if first.random_chip then new_chip+1;
run;
```

```
proc sql;
 create table final_sample as
 select  chip, location, grp, random_chip, random_num as random_order
 from step6;
 quit;
```

The final_sample list is ready for researchers to implement in the microarray experiment.

CONCLUSION

With the use of SAS macro language and Proc SQL, this tool implements an algorithm to randomize samples to the Illumina chips for microarray experiments. This makes it possible to avoid tedious manual work of trial and error in EXCEL. As a result, it greatly saves researchers' time and allows consistency due to the randomization algorithm. In addition, it can be applied to other types of experiments besides microarrays as well.

ACKNOWLEDGEMENTS

We would like to thank Esperanza Anguiano and Mamta Sharma from Institute of Immunology Research at Baylor Health Care System for providing this problem for which we developed the tool.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Huanying Qin
Enterprise: Baylor Institute of Immunology Research
Address: 3310 Live Oak, Suite 400, Dallas, Tx, 75204
Work Phone: 214-820-9064
E-mail: huanyinq@baylorhealth.edu