

SAS® and Clinical IVRS: Beyond Schedule Creation

Gayle Flynn, Cenduit, Durham, NC

ABSTRACT

SAS® is the preferred method for generating randomization and kit schedules used in clinical trials. The use of SAS® should not stop with the creation of the schedules, however, because it offers a great toolset to manage the risk of any IVR implementation. This paper will identify the leading causes of errors in IVR systems and provide practical solutions to eliminating the risk related to IVRS in the following areas:

1. Schedule handling
2. User Acceptance Testing
3. Managing risks prior to production
4. Managing risks post production

INTRODUCTION

IVR systems are used to screen, enroll and/or randomize subjects into clinical trials and assign study medication kits to those subjects entered into the system. These are very high risk systems, as small errors in randomization and/or kit assignment can jeopardize patient safety and/or study integrity.

Typically, IVR systems are developed in a compressed timeframe just prior to the start of the study during which the requirements, including the method of randomization and treatments assigned through the system may change due to factors outside the system, such as clinical supplies availability. These changes can lead to confusion and result in systems being built that do not adequately reflect the study requirements. In order to reduce the risk of potential problems and ensure that the systems are functioning as required, routine SAS checks should be performed on the data.

SCHEDULE HANLDLING

A great majority of IVR systems perform subject randomization via a permuted block randomization schedule. This schedule contains the random allocation of treatments that may potentially be assigned to subjects participating in a given clinical trial. Randomization schedules range from extremely simple, containing only a list of treatments, to highly complex where the list can be stratified by multiple factors.

In most cases, the randomization schedule is broken down into component blocks which are designed to facilitate the correct distribution of treatments. The blocksize is considered “unblinding” and must not be disclosed to investigators or other staff members as it may provide a mechanism for determining the treatment sequence, thus compromising the study’s validity.

During the course of the study each subject that is randomized will be assigned to one record in the randomization schedule based upon the randomization scheme. Each record within the schedule is associated with one unique randomization or sequence number in the schedule that is in turn associated with a give treatment regimen.

Once a subject has been randomized, the system assigns either a verbatim treatment or any number of treatment kit(s) to the subject. The interactive response technology system uses the production randomization schedule to perform subject randomizations during the course of the study.

An example of a randomization list is found below where BlockID defines each block and SectionID defines each stratification.

Sequence Number	Section ID	Block ID	Treatment Group	Treatment Description	Randomization Number
1	1	1	A	Drug 1	90001
2	1	1	B	Drug 2	90002
3	1	1	A	Drug 1	90003
4	1	1	A	Drug 2	90004

5	1	2	B	Drug 1	90005
6	1	2	A	Drug 2	90006
7	1	2	B	Drug 1	90007
8	1	2	A	Drug 2	90008
9	1	3	B	Drug 1	90009
10	1	3	A	Drug 2	90010
11	1	3	A	Drug 1	90011
12	1	3	B	Drug 2	90012

While performing adaptive randomization, IVR systems use an algorithm permitting the allocation of a treatment to a subject to be influenced by several factors. These factors include the current balance of allocated treatments along with the stratum to which the proposed subject belongs and the current balance within that existing stratum. If a system is performing adaptive randomization, it will not use a permuted block schedule, however; it may or may not use a table of random numbers for performing adaptive randomization.

A kit schedule will be used if study medication kits are assigned through the system regardless of the type of randomization that is being performed. The kit schedule is used to package the material by the clinical supplies provider as well as assign the kit by the system. The kit schedule must be maintained in a secure location as it is considered “unblinding.”

During the development of an interactive response system that is using permuted block schedules, two randomization and kit schedules (if required) are generated. The randomization schedules, although generated using different seeds, are identical in format. One schedule, commonly referred to as a “dummy” or “surrogate” schedule, is generated for use in the development and testing of the system. The “dummy” schedule is “unblinded” and allows the contents of the schedule to be utilized by the development team without compromising the blind of the study. The second schedule, referred to as the “production” schedule, is targeted for use in the production system. The production schedule must be kept in a secure environment as it is used for the purpose of randomizing subjects.

The management of the randomization and kit schedules is extremely important as confusion related to the identification of the dummy and production files may cause accidental unblinding or assignment of incorrect kits to subjects. This could potentially impact the results of the study or compromise subject safety. Additionally, if more than one set of randomization schedules is generated and sent to the provider of the interactive response system it must be ensured that the correct schedule is placed into the production environment.

SCHEDULE GENERATION VERIFICATION

At the time of schedule creation, a number of errors can arise with the potential to adversely affect the study or subject safety. Given the paramount importance of the randomization, a number of checks should be performed to ensure that the schedule has been created properly. The following checks should be implemented on the receipt of a randomization schedule and immediately prior to the schedule being loaded into the production environment.

1. A check to ensure that the schedule is created with the correct number of blocks.

If the schedule does not contain the anticipated number of blocks, follow up is required to determine if:

- the schedule being loaded is the correct “production” randomization schedule.
- the schedule being loaded has gone through some iterations with respect to requirements.
- the “dummy” schedule has been confused with the “production” schedule.
- the randomization schedule was inadvertently truncated.
- the schedule has been incorrectly generated.

Sample code for this check and output is found below. In this example, it was found that there were 10 blocks in the schedule.

```
proc sql;
create table count as
```

```
select count(distinct(block_id)) as Count, max(block_id) as Max, min(block_id) as Min
from duplicate_randomization;
quit;
```

Check to ensure the schedule is created with correct number of blocks

	Count	Max	Min
1	10	10	1

2. A check to ensure that each block contains the correct number of randomization records.

If the records within each block are not as anticipated, follow up is required to determine if:

- the schedule being loaded is the correct “production” randomization schedule.
- the schedule being loaded has gone through some iterations with respect to requirements.
- the “dummy” schedule has been confused with the “production” schedule.
- the randomization schedule was inadvertently truncated.
- the schedule has been incorrectly generated.

Sample code for this check and output is found below. In this example, a check was performed looking for four records inside each block. Blocks containing more than four records were output for investigation.

```
proc sql;
create table error as
select block_id, count(block_id) as Count
from duplicate_randomization
group by block_id
having count ne 4;
quit;
```

Check to ensure each block contains correct number of randomization records

Obs	Block_ID	Count
1	4	5

3. A check to ensure that the treatment ratios are correct.

If the treatment ratios are not as anticipated, follow up is required to determine if the schedule being loaded has gone through some iterations with respect to requirements, or if the schedule has been incorrectly generated.

Sample code for this check and output is found below. In this example, a check was performed looking for two of each type of treatment inside each block. Blocks containing records without two treatment groups were output for investigation.

```
proc sql;
create table error_treatment_group as
select block_id, treatment_group, count(treatment_group) as Count
from duplicate_randomization
group by block_id, treatment_group
having count ne 2;
quit;
```

Check to see that the treatment ratios are correct

Obs	Block_ID	Treatment_Group	Count
1	1	A	3
2	1	B	1
3	4	B	3

4. A check to ensure that the treatment group descriptions are correct.

If the number of treatment group descriptions are not as anticipated follow up is required to determine if the schedule being loaded has gone through some iterations with respect to requirements, or if the schedule has been incorrectly generated.

Sample code for this check and output is found below. In this example, a check was performed looking for two of each type of treatment description inside each block. Blocks containing more than two treatment descriptions were output for investigation.

```
proc sql;
create table error_treatment_description as
select block_id,treatment_description , count(treatment_description) as Count
from duplicate_randomization
group by block_id, treatment_description
having count ne 2;
quit;
```

Check to ensure treatment descriptions are correct

Obs	Block_ID	Treatment_Description	Count
1	4	Drug 1	3

5. A check to ensure that the mapping between treatment_group and treatment group descriptions are correct.

The presence of treatment mis-mapping in a file may potentially impact patient safety in a clinical trial. This check should be performed with all data present in the system at each load. Sample code for this check and output is found below. In this example both Treatment Group "A" and "B" map incorrectly to both Drug 1 and Drug 2 rather than Treatment Group "A" being associated with "Drug 1" Treatment Group "B" being associated with "Drug 2".

```
proc sql;
create table mapping as
select distinct(treatment_group), treatment_description
from duplicate_randomization;
quit;
```

Treatment Group and Description Mapping

Obs	Treatment_Group	Treatment_Description
1	A	Drug 1
2	A	Drug 2
3	B	Drug 1
4	B	Drug 2

6. A check to ensure that each randomization number is correct with regard to format.
7. A check to ensure that each randomization number is unique.

The presence of duplicate randomization numbers will impact system behavior which may potentially impact patient safety. This check should be performed with all data present in the system at each load. Sample code for this check and output is found below. In this example there are three observations with a duplicate randomization number which must be resolved prior to loading the schedule.

```
proc sort data=duplicate_randomization nodupkey dupout=error_randomization;
by Randomization_Number;
run;
```

Proc Sort Nodupkey On Randomization Number

Obs	Sequence_Number	Section_ID	Block_ID	Treatment_Group	Treatment_Description	Randomization_Number
1	13	1	4	B	Drug 1	90013
2	37	1	10	A	Drug 1	90037
3	38	1	10	B	Drug 2	90038

8. A check to understand the sequencing of the randomization schedule.

The last check is to verify that the sequencing is correct within the numbers on the schedule. In the example below, any numbers determined to be out of sequence are output to a dataset for review.

```
proc sort data=Duplicate_Randomization;
by randomization_number;
run;
data Out_Sequence(drop=flag sequence);
set Duplicate_Randomization;
if _n_ = 1 or flag=1 then do;
sequence = randomization_number;
flag = 0;
end;

if sequence ne randomization_number then do;
output;
```

```

flag=1;
end;

sequence = sequence + 1;
retain sequence flag;
run;

```

Out Of Sequence

Obs	Sequence_Number	Section_ID	Block_ID	Treatment_Group	Treatment_Description	Randomization_Number	Age	Subject_Number
1	13	1	4	B	Drug 1	90013	31	114
2	37	1	10	A	Drug 1	90037	69	140
3	38	1	10	B	Drug 2	90038	15	141

In addition to the randomization, the following items should be checked on the kit schedule on receipt and immediately prior to the schedule being loaded into the production environment.

1. A check to ensure that the schedule is created with the correct number of records.
2. A check to ensure that each kit number is correct with regard to format.
3. A check to ensure that each kit number is unique.

Schedule Loading

It is critical to ensure that the randomization schedule that has been loaded into the production environment is the “production” schedule rather than the “dummy” schedule. Accidental unblinding of UAT and project team members and reviewers of system documentation may result if the “dummy” schedule is inadvertently loaded into production or if the “production” schedule was accidentally loaded into the development or test environment. Additionally if an incorrect kit schedule is loaded into the production system, the system will assign kits in a manner not consistent with the randomization schedule. This can impact study integrity and patient safety.

During the development of the system, study requirements may change frequently, causing more than one set of randomization and /or kit schedules to be generated. In this case an erroneous schedule may be placed into production due to poor schedule handling procedures. A comparison of the schedule loaded into production against the schedule that is housed by the randomization statistician should be performed so it is clear that no errors have occurred during the handling of the schedules. The following checks should be performed to ensure that the schedules are handled appropriately:

- A check that the randomization schedule slated for production is not identical to the randomization schedule that was used for development and testing purposes.
- A check to ensure that the kit schedule is not identical to the kit schedule that was used for development and testing purposes.
- A check on the schedule that was supplied to the clinical supplies provider to ensure that it is consistent.

PROC COMPARE

The easiest way to ensure that the schedules are identical where they are expected to be as in the case of the kit schedules between the clinical supplies provider and the interactive response provider or different as in the case between the dummy and production randomization and kit schedules is to perform a PROC COMPARE.

The output generated is listed below. In this case the PROC COMPARE indicated that the two schedules were different.

Observation Summary

Observation	Base	Compare
First Obs	1	1
First Unequal	1	1
Last Unequal	28	28
Last Obs	40	40

Number of Observations in Common: 40.
 Total Number of Observations Read from WORK.PRODUCTION: 40.
 Total Number of Observations Read from WORK.DUMMY: 40.

Number of Observations with Some Compared Variables Unequal:

15.

Number of Observations with All Compared Variables Equal: 25.

Values Comparison Summary

Number of Variables Compared with All Observations Equal: 5.
 Number of Variables Compared with Some Observations Unequal:

1.

Total Number of Values which Compare Unequal: 15.
 Maximum Difference: 0.

APPENDED SCHEDULES

As stated above, each randomization number must be unique within the schedule. It may be necessary to append a new schedule to the original schedule to support randomization of the study. If this occurs, extreme care must be taken to ensure that the new schedule does not contain duplicate randomization numbers or block numbers. Additionally, the new schedule must be checked to ensure that the treatment description has not become inadvertently crossed and the treatment ratios remain consistent between the two schedules.

In order to perform this check, the two schedules must be concatenated for purposes of comparison. This check should be performed prior to loading the subsequent schedule into production. The correctness of the subsequent schedule can be easily checked by concatenating it to the previous schedules and performing the same checks as indicated above.

USER ACCEPTANCE TESTING

User acceptance testing is conducted to ensure that the system has been created to specification. Although timelines are tight, this step should not be taken lightly. A sufficient number of subjects are randomized in the test system to ensure that it is operating in accordance with study documentation. As mentioned above, the system may use a randomization schedule for permuted block randomizations or generate the randomization if the system is designed to use some type of adaptive randomization such as Simon and Pocock.

The randomization of subjects in UAT should follow the randomization and ensure that multiple sites and stratifications are used in a non sequential fashion. The randomization and kit schedules may be analyzed using SAS to ensure that the system is behaving as anticipated.

MANAGING RISKS DURING THE STUDY

There are a number of risks that must be managed during the maintenance phase of the interactive response system. Once the system is in production, conducting periodic checks is an effective way to manage these risks. This should include:

1. EDIT RANGE CHECKS

A check of the database while the study is in production can provide assurances that the required edit checks for age and other demographic factors are being performed properly. This check ensures that any eligibility criteria are verified and that the system assigns randomization and kit numbers consistent with what is required for the study. In the example below, the code checks to see if the age of entered subjects is outside the range for randomizing subjects.

```
proc sql;
select * from Duplicate_Randomization
where age < 18 or age > 65;
quit;
```

Age Range Check

Sequence Number	Section ID	Block ID	Treatment Group	Treatment Description	Randomization Number	Age	Subject Number
37	1	10	A	Drug 1	90037	69	140
38	1	10	B	Drug 2	90038	15	141

2. RANDOMIZATION CHECKS

Immediately after go live, the first check that should take place on the data contained in the system is to ensure that the date the first subject has been screened, enrolled or randomized into the system is greater than the date/time that the system has been set live. This check will guard against any inadvertent data from system testing being placed into the production system and impacting the randomization of subjects into the study.

After several subjects have been randomized, a check should be performed to verify the sequencing of the assignment of randomization records. This check ensures that subjects are stratified correctly and that the system is following the randomization scheme as indicated. For studies using a central randomization scheme, there should be no skipping of individual randomization records as this type of randomization calls for the sequential assignment of randomization records. If the system is using any type of stratified randomization, then the assignment of records must be sequential within each stratum. This means that there will be skipping of randomization records between the strata but not within any given stratum as identified by the SectionID. The data should be sorted by SectionID, Site ID, if the study is stratified by site, and randomization date, for studies that are stratified prior to performing the check. The data need only be sorted by randomization date for non-stratified studies, as the assignment of records will be performed sequentially. A check should also be implemented to ensure that subjects are randomized to the correct stratum based upon the data stored, so that the system is not impacted by errors in stratification mapping.

The overall system functioning of the interactive response system also needs to be checked to ensure that there are no assignments of duplicate subject, randomization or kit numbers. Errors may occur at the time when more than one subject is attempting to be randomized concurrently. Although unlikely, some randomization systems will not enforce uniqueness which results in this type of error.

For studies using Pocock and Simon, periodic checks should be implemented to check on the balance of the randomization.

3. CAPPING CHECKS

Another important check is to ensure that the system capping has been applied appropriately with respect to site, stratum, country or study related caps. A simple check of the count of the number of subjects within each cap against the target levels for the cap is sufficient.

4. CONSISTENCY CHECKS

A number of checks must be put in place to ensure that the system is performing in a manner consistent with its design. If the system is using a kit list to assign the study medication it is useful to identify subjects who were randomized but did not have a corresponding kit assigned within the system. This mismatch may be caused by the system assigning the randomization record despite no kits being available at the site, or could indicate that the kit or randomization record has been updated manually in the system. Sample code and output can be found below.

```
proc sql;
select * from Duplicate_Randomization where subject_number not in (select subject_number from
kit_schedule)
and subject_number ne .;
quit;
```

Subjects in the Rand Schedule but not in Kit Schedule

Sequence Number	Section ID	Block ID	Treatment Group	Treatment Description	Randomization Number	Age	Subject Number
4	1	1	A	Drug 2	90004	21	204

A check should also be incorporated to determine if kits were assigned to subjects who were not randomized in the system. This assignment of a kit may have been the result of a manual update to the system or a system error.

Additional checks should also be included to determine what subjects have been randomized but assigned a treatment other than what they were randomized to. Once a mismatch has been identified, an investigation must be performed to see if the kit record in question was updated for the particular subject (the mismatch was caused by an investigational site error) or if it was assigned in error by the system, in which case, the system is not functioning as intended.

5. EXPIRY DATE CHECKS

Another associated with IVR systems is that kits that are very close to or past the expiration date for the visit window in the system are assigned to subjects. The assignment of treatment kits within the system usually requires that the kits will not expire during the period between assignment to the subject and the subject's next visit. This error could be the result of the system not incorporating the required checks on expiry date, simplistic calculations on the expiry date that do not take into account multiple kits being dispensed at one visit, or the manual assignment of the kits outside of the system.

CONCLUSION

In conclusion, SAS is a valuable tool for managing risk of interactive response systems through the performance of a series of predefined checks. SAS also allows facilitates easy verification of file contents to help prevent the introduction of errors outside of the system.

CONTACT INFORMATION

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

Gayle Flynn
Cenduit, LLC
1007 Slater Road, Suite 301
Durham, NC 27703
Work Phone: 919-998-3868
Fax: 919-998-3899
E-mail: gayle.flynn@cenduit.com
Web: www.cenduit.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
Other brand and product names are trademarks of their respective companies.