Paper 5112-2020

Assessing Performance of Risk-based Testing

Amber Randall, SCHARP − Fred Hutchinson Cancer Research Center

William Coar, Axio Research − A Cytel Company

ABSTRACT

Trends in the regulatory landscape point to risk-based approaches to ensure high quality data and reporting for clinical trials. Risk-based methods for validation of production programming code which assign testing methods of varying robustness based on an assessment of risk have been evaluated and accepted by some industry leaders, yet they have not been fully adopted. Some view risk-based testing as simply an attempt to save money or compensate for limited resources while claiming a minimal impact on overall quality. While that may sometimes be the case, the intent should rather be to focus finite resources on what matters most.

Even with the robust gold standard of full independent reproduction, mistakes still happen. Humans make errors. Therefore, risk and consequence should be considered in choosing verification methods with a resource emphasis on those areas with greatest impact. However, the assessment of these decisions must be regularly and consistently evaluated to ensure that they are appropriate and effective.

Metrics both within and across projects can be implemented to aid in this evaluation. They can report the incidence, type, and method of identification of issues found at various timepoints in the production process. This includes issues found internally prior to the completion of output verification (i.e., during testing), internally during final package review, and during external review. These data are crucial for the effective evaluation of the performance of risk-based testing methods and decisions.

INTRODUCTION

As in many industries, clinical research teams often find themselves having to do more work with fewer resources while still maintaining a high level of quality. This includes data collection and monitoring as well as data analysis and statistical programming. In response, there has been an industry-wide movement toward risk-based approaches to quality. While a gain in efficiency can lead to a cost savings, this is not the primary motivation for this approach. The real goal is to maintain a high level of quality with a fixed number of resources by focusing efforts in areas with the highest impact and risk of error. The United States Food and Drug Administration has already provided guidance on a risk-based approach to monitoring of clinical trial data collection.[1] Other functional groups within clinical research, such as statistical programming, have continued the conversation.

We define *validation* as the collection of efforts made to ensure and document that production programs produce expected results and, in this paper, we use this synonymously with the terms *verification* and *testing*. We acknowledge that some organizations might differentiate between these terms. Verification might be as simple as a peer review of code or reviewing for expected results, or as involved as complete and independent reproduction of the production results. These differ from a formal validation approach that might be taken for software which we feel does not directly apply to the verification of output from SAS® or other statistical programs.

The purpose of validation of any statistical programming is simple: important decisions are being made based on results of analyses generated by those statistical programs. In some

cases where companies must comply with regulatory requirements, validation may even be mandated. This is true for the pharmaceutical industry. There are many approaches to validating a statistical program, or verifying the resulting output, but not all employ the same degree of rigor and thoroughness. Of course, the current gold standard of full independent reproduction is imperative for the assessment of a primary endpoint or key safety data, however, some results may have very little impact on important decision making or perhaps the risk of an error occurring in certain programs is extremely low. In these examples, the cost of full independent reproduction is not representative of the small incremental benefit. See Figure 1. It is our experience that while an increased amount of effort at the beginning of a programming process can initially lead to large gains in quality, it is not a linear relationship. At some point increasing effort does not continue to provide tangible benefit. Perhaps less labor-intensive methods may be employed for some tasks without a notable impact on the overall interpretability of the report.

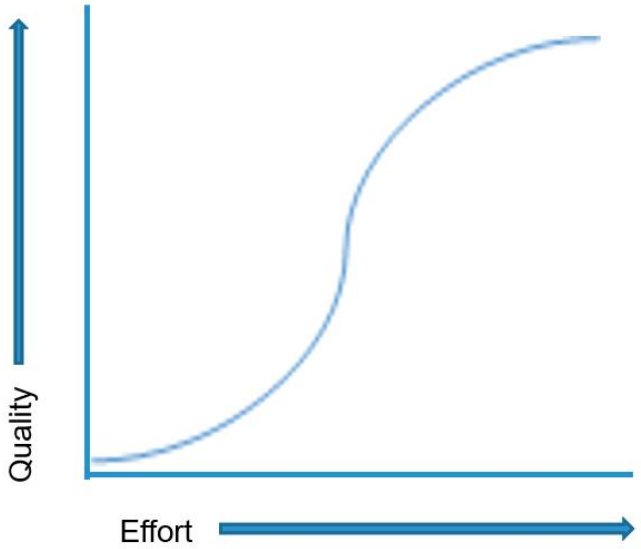

Figure 1. An Illustration of the Relationship between Effort and Quality in Validation

A high level of overall accuracy can be maintained and efficiency increased if, rather than independently reprogramming every element of a report, an alternative, yet appropriate, element-specific method of validation is chosen that is commensurate with the risk and potential impact of an error, as assessed by a qualified team member. This is the basis for risk-based validation approaches.

Acceptance of risk-based testing has already been established. Several recent presentations such as Marrer 2019 and Randall et al. in 2018 have also addressed the topic. Other papers exist suggesting that risk-based validation is not new. In fact, in our professional experience many organizations already subscribe to some form of risk-based validation and allow for the application of different verification methods. Examples include independent code review for programs that simply list data, or using corresponding summary table output to visually verify a figure rather than fully reproducing it.

In order for risk-based validation to prove effective in the long term, metrics on errors and causes of failure must be thoughtfully collected and analyzed. This is important for the identification of steps in the programming and verification process with increased potential for failure such as not effectively assessing risk or not assigning an appropriate validation method. Patterns may also suggest systematic errors in the existing programming or validation process such as not handling dirty or unexpected data appropriately or highlighting gaps in training. However, even if no obvious patterns exist, data captured for metrics may still provide important information for future prevention of critical errors.

Risk-based validation methods can take on many forms, some more detailed than others. Have you ever opted to do a tertiary review of a primary endpoint? Have you ever discussed how to do a structured review of code for importing raw data from different sources or how to compare datasets in different types of file formats? Most of us at some time have made a risk-based decision during validation of a SAS program, though it may not have been formally defined as such. This is often driven by individual expectations, company SOPs, industry standards, personal historical experience or project knowledge. Many factors must be considered in the development of a risk-based validation system that is appropriate for your project, process and organization. As such, our aim here is not to provide a step-by-step guide in how to build your validation system or specifically which metrics to collect, but rather to discuss key points to consider as you assess where a risk-based approach and targeted metrics collection might help in your individualized process. Every programming process has unique considerations and validation strategies must reflect that.

A RISK-BASED VALIDATION APPROACH

As described in [3], a simple risk-based validation approach can be designed with 7 steps. For a more in-depth discussion of these steps, please refer to this paper. A brief summary is provided here.

STEP 1: DEFINE YOUR PROGRAMMING PROCESS

It is very important to first map key steps in your programming process from report request through delivery to identify areas where a risk assessment should or can be made and to think about what types of verification options would be possible and appropriate at each validation step. For example, early steps may require conversion of non-SAS data to SAS data prior to analysis. Depending on the size of the data, format of the data, and required manipulations, validation may simply consist of comparing attributes (such as the number columns and rows, variable attributes) with corresponding information or metadata that may have come with the transfer. Later analysis steps, however, will likely require data manipulations for a more complex result and will thus also require a far more rigorous validation approach.

STEP 2: DEFINE THE DELIVERABLE COMPONENTS

A clear and concise programming tracking sheet or program inventory is necessary to document programming tasks and decisions. Every element of production output comprising a final deliverable package should be included here. These might include analysis datasets, statistical tables and figures, and data listings. At a minimum for each element, source data, file names of associated programs and outputs, names of production programmers and validation programmers, and dates of completion should be noted.

For each program, the programming lead should also note the type of execution for that particular run. The purpose of defining the execution type is to assist with the assignment of risk level as described in the next section, as well as clarifying expectations with respect to the required specifications for each program. These may include:

- Run of new dataset
- Run of new report element (i.e., summary table, figure, or listing)
- Re-run of existing dataset program w/o modification
- Re-run of existing dataset program with modification
- Re-run of existing report element program w/o modification
- Re-run of existing report element program with modification

You might also consider noting whether you expect changes in the format or structure of

input data or whether a formal comparison of old and new iterations of data cuts has been performed and if so, any relevant outcome from the comparison. This information can also provide insight into the risk associated with any given report element.

STEP 3: ASSIGN A RISK LEVEL FOR EACH ITEM

An assessment of risk must be performed for each program to guide the selection of an appropriate verification method. Qualified team members with a thorough level of knowledge of the project, the data, and the programming efforts for this task will likely be senior-level statisticians or statistical programmers, but input from the entire programming team should be considered. In order to effectively assess risk, a thoughtful and critical approach is necessary.

Without going into a detailed discussion, we define risk as a function of three components:

- Overall impact on the interpretability of the deliverable
- Complexity of the underlying data
- Likelihood of programming error

Others may view risk differently, which is expected in a risk-based approach. After all, there will be subjectivity when it comes to "acceptable" risk. Risk assessment may be a numerical calculation, it may be guided by a summary table, or it may be a function of individual experience.

Regardless of how your team assesses risk, it should ultimately fall into tiered categories. For simplicity we suggest three (low/moderate/high). To standardize your approach across your team, identify key factors that might lead to particular risk assignments. It is important to define and collect information on factors that contribute to your risk assessment. More discussion on this is presented in Step 5.

STEP 4: CHOOSE A VALIDATION METHOD THAT IS APPROPRIATE FOR THE RISK LEVEL

It is best to begin by pre-defining a list of acceptable validation methods as well as outline considerations for implementation of each given the different levels of risk. This should include a general guidance on how to apply and perform each one as well as which would be most appropriate for each risk level. The assigned validation method or methods should be clearly documented in the tracker and assigned to a validation programmer within your tracking sheet. Common validation methods include, but are not limited to, the following:

- Full independent reproduction
- Independent code review
- Review for consistency within and between reports
- Review for expected changes from previous output
- Manual visual comparison against source or other output
- Visual review of cosmetic changes
- Self-review

As mentioned in Step 4, assessment of risk may be a function of individual experience. In practice so can the assignment of a verification method. For example, we often re-run a previously validated program on an updated set of data. Experience might tell us that if we can verify the structure and format of our input data and our program has not changed, then full verification of the program output might not be required. Perhaps a review of program logs is sufficient. Or, when data are mature and we know what changes in the

output are expected, then a review for the expected changes only, rather than the entire output, may be sufficient.

STEP 5: DOCUMENT YOUR PROCESS AND DECISIONS

Even though we have delayed formal discussion of documentation until Step 5, it should be thoroughly maintained throughout the risk-based validation process. All critical steps and decisions should be recorded both for future assessment of effectiveness, but also as justification for decisions if questions arise. Your approach for maintaining documentation should be well defined and mandated for your team and performed within a controlled environment if possible.

As mentioned above a tracker should minimally contain the following:

- List of items to be programmed
- Source data
- Names of production and validation programs
- Assigned production programmer and date of program
- Assigned validation programmer and date of validation
- Outcome of the validation
- if an issue is noted in validation, the process for resolution

A well-designed tracker for a risk-based validation approach should be more robust. To evaluate your (risk-based) validation strategy, we propose starting small, and adding appropriate fields as you learn. For example, initial data for reporting on metrics might be:

- Type of execution (as discussed in Step 2)
- Assessed risk
- Assigned validation method
- Outcome of validation
- Outcome of internal review
- Outcome of external review
- Categorical causes of errors at each process check point

Note that simple metrics can be obtained for each of the bullets above as long as information is collected consistently and in a controlled manner. For example, if Excel is used as the tracking system, pre-defined drop-downs should be utilized so that data can be easily categorized and summarized. Tools and considerations for collecting metrics will be further discussed below.

STEP 6: PERFORM THE VALIDATION

If your documentation − specifications and tracker − has been meticulously completed, then the validation team should find clear directions and expectations to complete their tasks. The validation programmer should follow instructions and document the outcome of each validation. If issues are identified, they should be recorded and sent back to the programming team or programming lead for review, assessment and resolution.

STEP 7: REVIEW YOUR METRICS AND REASSESS PROCESS DECISIONS

One of the most important steps of risk-based validation, and the focus of this paper, is evaluation of the effectiveness of the process. A risk-based validation system is defensible only when you have proof that it is working as intended. Issues with not only the

assessments of risk and assignments of validation methods, but also systematic problems in your programming procedures and practices can be elucidated. Detailed, real time issue tracking, recording, and classification of errors identified both before and after delivery is key. It is important to retain all of this information for assessment, rather than just a record that says a program passed validation. These data can be mined for a wealth of information about the effectiveness of your process and proficiency of your team. This gets to the heart of our discussion and will be elaborated below.

WHY DO METRICS MATTER?

Metrics provide a window into whether or not the programming and validation systems built by your team are being applied as intended and working as expected. If errors are slipping through your standard process and being found by management review, or worse yet, after delivery to a client, your team should have the data and power to look at where the system is breaking down.

A well-built programming system and especially a risk-based validation system should be modified when there is evidence that the results are not to the quality expected or if the parameters of the programming request have changed. Clinical trials are dynamic and resources are finite. Programming teams need to be able to focus efforts where they matter most and be confident in their approach. However, if metrics suggest that the approach is unintentionally decreasing overall quality, then the approach must change.

Iterative deliverables, common in clinical trial programming, often change as a study or project matures. What is most important or error prone at an early iteration may be different than what is important or error prone at a later iteration. An example of this is the production of reports for data monitoring committees [4]. The most important and meaningful content of the report often changes as the study progresses. While medical history and demographics may be of importance during the enrollment period at the beginning of the study, careful analysis of endpoints will inevitably become more important near the end. Perhaps there have been data updates, safety concerns or special requests since the last report run. Maybe your programming team noted something in the data early on that could affect a future iteration. Maybe limited data exist early in the study and it is not possible to even generate all required outputs from the onset. It is important in this situation to review the risk assessment and validation method for each item at each subsequent iteration and adjust as necessary to reflect the new levels of risk at each delivery. You will likely not validate the first report with the same approach as the last report. Metrics should be reviewed along the way to make sure your assessments and process are working and continuously adjust as necessary.

In addition to your risk assessment and validation method assignment algorithms, metrics collected during the review of your programming and validation process may also identify the need for systematic changes to your standard programming procedures or the need for additional training in the application of your standard system. Remember, information must be retained throughout the process (in an issue log) to better understand systematic problems.

For example, insufficient detail in specifications may result in multiple rounds of issue identification and resolution. Conversely, too much detail may inspire complacent validation. In the former, multiple interpretations of ambiguous instructions can lead to discrepancies that must be resolved. In the later, the lack of necessary critical thinking resulting from overly detailed specifications may lead to errors that occur because the validation code too closely resembles the production code and neither programmer questioned whether the specification actually made sense. We discuss this more in the Business Case section below on Operations. Well documented metrics on issue identification may help to address problems with standard procedures.

Beyond looking at individual projects, your metrics assessments should also include a more global examination of the errors found across several projects in your organization. You need to consider how you will analyze metrics across projects and consider the most pressing questions when building your programming and validation system.

THE BUSINESS CASE FOR METRICS EVALUATION

In addition to the ideal of high quality, several business cases for the collection and evaluation of programming and validation metrics can be argued. While quality for quality's sake is good – for example, it supports the efficient use of limited resources and ensures patient and participant safety, from a pure business perspective it is important for maintaining a good reputation in the industry and protecting your programming team from the burden of excessive unexpected challenges.

CLIENT/PARTNER RELATIONS

First and foremost, evaluating metrics and assessing your programming process is important for maintaining client and partner confidence in the work that your team is doing. If a client frequently finds errors in a validated report, it is going to both cast doubt on your skills and process, but also raise suspicion of any future deliverables programmed by your team. At some point your organization's ability to secure more work may be impacted. Without having data collected for metrics, it may be difficult to determine if such errors are associated with inaccurate risk assessment, one or more particular validation methods, or even individual personnel.

Beyond simply addressing quality, metrics can provide insights into realistic estimates for how long work should take. This is useful information for drafting statements of work and contracts. Additionally, metrics may be able to tell you why work for certain clients is taking long than others. Maybe you are receiving more dirty, unexpected, or inconsistent data from these clients. Maybe they are requesting more changes. Any change increases risk of error and must be factored into your process planning. An increased risk may suggest more robust validation methods which can impact resource requirements. Metrics can be further used when estimating both project cost and timelines as well as resource planning.

A Note about Time-Based Metrics

We admit that time-based metrics can be challenging to collect, predominantly due to the difficulty in gathering accurate data. It is much easier to make general statements about time, such as "it should take a few days, yet it seems to be taking 2 weeks", but harder to monitor and collect actual hours spent per element. Our world is very dynamic, multitasking is the norm, and programming effort does not always impact just one programming unit. It may be a more reasonable approach to look at time to specific milestones within any given deliverable or project rather than by element.

OPERATIONS

Evaluating metrics from your programming and validation process can elucidate operational components that are inefficient or need reexamination. This could obviously be programming and validation infrastructure and best practices, but can also include methods of team or cross-team communication, documentation, or even workflow planning and triage.

Metrics can give you focus points for internal standard process review. Metrics can also provide insight into your specification strategy. How much detail is really necessary in your specification? Not enough detail may allow for too much individual interpretation and lead to more discrepancies. Too much detail may mean that your programmers don't really need to think critically about the data or intent. This might mean that the programmer and validator

are getting the same result based on overly detailed instructions, but they may not be considering whether the specifications are actually reasonable or correct. This can lead to more errors found by management or the client even though everything tested clean as per your process. How much should the tester be forced to think critically about the requirements of the analysis plan and data? If you find that your issues are related to the writing of the specs, how do you improve them? What are the minimum specifications for the skeptical reviewer? Specification detail should be considered when evaluating risk and determining an appropriate validation method.

Metrics can give insight into appropriate resourcing for a given project. As mentioned at the beginning, resources are often finite and more robust efforts require more resources. However, increasing resources beyond a certain level may only provide small incremental gains in quality if the additional effort is not focused appropriately. See Figure 1. Finding the balance between effort and quality is challenging, but metrics can provide guidance.

In another example, suppose your analytics team is receiving data from an internal data management team, metrics may help identify gaps in your internal data quality process. Are more errors being found in the validation process because of dirty or unexpected data? Why are the data not clean? Is the data management team getting alerts when data issues are found? Are they being addressed promptly? If not, why not? Is your programming team writing defensive data-driven code to identify or handle unexpected or even expected, but not yet collected data? If you are receiving dirtier-than-normal data or data in a new structure or format from a client that is negatively affecting the performance of your analytics team, is there a contractual protection for you? Is that something that you should consider in the future? Can your team develop programmatic tools to evaluate data quality and consistency between transfers so that your team is not caught off guard?

As we ponder all of these questions, it becomes clear that any one of them may change an assessment of risk (and subsequently the assigned validation methods). In this last example, metrics can provide insights not just on your strategy of risk assessment, but also lead to improvements with your internal data management process.

Finally, and back to the original point of this paper, metrics can tell you whether your risk-based validation system is working. Are your efforts focused where they really matter? Are important errors being identified prior to a deliverable leaving the hands of your programming team? The metrics that you collect during your process should help you assess whether the level of risk that was chosen for each programming element was accurate and whether the method of validation chosen was appropriate.

TRAINING

Metrics can also help guide decisions regarding training. If errors are commonly associated with one or two members of your programming team, it may mean that you need to provide retraining on the validation system for those members. Or, it may signal the need for a skills assessment such as the ability to apply defensive, data-driven programming practices. It may make it apparent that programmers are not testing to the specification, but rather just trying to match results with the lead programmer. Metrics can also help identify the right time to introduce new concepts to your team or reinforce old ones.

WHAT ARE THE GOALS OF AN ASSESSMENT OF VALIDATION METRICS?

Your organization may define many goals for the evaluation of your validation metrics, but at a high level, we would suggest the following three. First, minimize the errors that slip through your core programming process and are found by individuals outside of your direct team. If errors exist, make sure it is your team that finds them. Second, refine the efficiency and accuracy of your risk-based validation process. If you have implemented risk-

based validation, make sure it is working and that you are taking advantage of the efficiencies it can offer. Third, identify amendments to your established process to eliminate systematic or frequently occurring errors. Are there operational or infrastructure improvements that can reduce errors?

MINIMIZE ERRORS THAT SLIP THROUGH YOUR PROCESS

If errors in programming exist, then they should be identified by your team rather than during senior manager review, or worse by client review. If errors are escaping your process, you need to find out why. Does the problem stem from an insufficient process or an incorrect implementation of a sufficient process? Were the instructions to the team clear? Was training implemented sufficiently? Was your risk-assessment accurate? Were the validation methods appropriate? Are issues being caused by factors outside of your control such as dirty or unexpected data? What can be done to mitigate this? It may not ultimately be related to the risk assessment, but without metrics, we can only speculate.

REFINE THE EFFICIENCY OF YOUR RISK-BASED VALIDATION SYSTEM

So, your team has designed a risk-based validation system, now you need to know whether or not it is working as expected and whether you are realizing the efficiencies that it should deliver. Is there a way to measure whether your validation times have decreased in low risk areas, yet increased in high risk areas? Are you really focusing on what matters most or is your team falling into old habits such as double programming everything without taking a thoughtful approach? We have experienced programming teams spending hours or even days verifying derived variables that are not ultimately even reported. That time can be spent more meaningfully elsewhere with the proper thoughtful planning.

Resources and time are ultimately fixed. At some point, decisions will be made to place greater effort in some areas, and less in others. Keep in mind that lower effort should NOT imply lower quality in a risk-based setting, but you must be able to show that your assessments are accurate and effective. Is your team able to reduce the need to double program everything and still provide a high-quality deliverable with less stress and overwork?

IDENTIFY GAPS IN OR AMENDMENTS TO YOUR PROCESS

Your metrics assessment should also be equipped identify systematic errors in your programming process. Should your philosophy toward specifications be reexamined? Should your programming infrastructure be modified to automate or dictate more best practices or strategies? Are there certain skill sets or training modules that are lacking within your team? Is more defensive programming needed? Standard process can always be improved.

HOW DO WE DEFINE USEFUL METRICS?

In order for any metrics to be useful, it is necessary to think about what questions you really want to answer and then define meaningful classifications and categories of issues that might address them. These will be specific to your organization, project, team and/or process but may include some or all of the following:

WHO FOUND THE ERROR AND WHEN?

A member of your team during routine validation? A member of the management team during final review prior to external delivery? A client or external reviewer during a formal review process? A committee member during a formal meeting?

HOW WAS THE ERROR IDENTIFIED?

By your routine validation process? By automated electronic validation? By manual comparison? By third party independent review? During cross-table checks? By comparing

to old reports? Was your chosen validation procedure appropriate?

IF NOT FOUND IN THE ROUTINE VALIDATION PROCESS, WHY NOT?

Ineffective process? Incomplete process? Insufficient training or understanding of the process? Insufficient understanding of the study or task? Human error? Insufficient time or resources? Insufficient skill or experience?

WHAT WAS THE IMPACT OF THE ERROR?

No impact? Delayed delivery? Inappropriate decision made by the study team or independent committee? Patient safety issue? Embarrassment? Lack of client confidence? Loss of future business?

DOES YOUR PROCESS SHORTEN DELIVERY TIME?

What was the time to delivery? How long does it take to program and validate each element or milestone? Did an error delay a delivery? Are your contract or statements of work estimates realistic? Is your risk-based validation process working as expected, i.e. are you realizing efficiency benefits?

WHAT WAS THE UNDERLYING CAUSE OF THE ERROR?

Incomplete or confusing specification? Mixed interpretation of specification? Unexpected or dirty data? Unexpected change in source data content or format? Poor understanding of the protocol or analysis expectations? Poor understanding of process? Systematic or infrastructure weakness? Gap in programmer skillset? Insufficient resourcing?

IS THE ERROR RELATED TO DIRTY OR UNEXPECTED DATA?

Does the data cleaning process need to change? If data issues are found, is the message getting back to the data management team? Are queries dealt with promptly? Can the programmers write more defensive code?

HOW DO YOU COLLECT AND TRACK METRICS - CONSIDERATIONS FOR MAINTAINING AN EFFECTIVE METRICS LOG?

As mentioned above in Risk-based Validation Method Step 5, documentation is critical for building and maintaining a successful process. Without this, metrics will not be readily available, and we are left with assessing errors on a case-by-case basis rather than having the ability to look for trends.

An element tracking sheet can work well for collecting metrics during the routine programming process, but you should be proactive and insightful enough to pre-define the metrics collected so that appropriate and useful information is captured throughout the process. Drop down menus with predefined options for categories can work well if using a spreadsheet. There should be a balance between collecting enough information to be useful and collecting so much that it is burdensome. Excel can work for this purpose but is imperfect since it offers no audit trail or data entry permissions. Furthermore, Excel is not particularly useful for analyzing global issue tracking across projects. Historically, there have not been many off-the-shelf software options designed to support process management needs of statistical programmers in the pharmaceutical industry. Some companies have chosen the rather expensive route of in-house development, but that may not be an option for smaller companies. Fortunately, some new commercial options are in development and we look forward to learning how they function in practice.

While the programming tracking sheet for any deliverable may work well for recording errors found in the regular programming process, it may not work well for errors found after the programming process – such as those identified by management review or after report

delivery. This information needs to be communicated back to the programming team in order to be tracked. Your team should think closely about collection strategies that work for them. If available in your organization, you may be able to utilize the incident tracking feature of a Quality Management System for errors found outside of your programming team.

You should also consider how to organize issues collected for a particular deliverable versus iterative deliverables.

One more consideration to note when strategizing on collection methods is reporting bias. We recognize that some staff may be reluctant to accurately record errors if they feel that these metrics may be used against them in a formal assessment of performance. We caution against this practice. In a trusted and quality-driven environment, staff should feel comfortable to honestly report what happens when it happens. Bias should not be a reason to ignore metrics. If management wants and effective reporting process, then they need to create a supportive, safe environment.

HOW DO WE EVALUATE OUR METRICS?

We all have our favorite ways to work with data. While some programmers are comfortable working in Excel, others prefer to work within SAS or other statistical software. If we are thoughtful about the questions we want to answer and collect appropriate information during the programming process, then evaluating metrics can be straightforward.

Simple tabulations and graphics can be used tell the story. There is no need to look at inferential statistics or statistical modeling. Understand that reporting bias is real, and will exist when recording issues and errors. You should think about points where collection bias may be likely and adjust your approach or interpretation accordingly. Accuracy of reporting of issues and data collection should be considered. Even if you know that the magnitude of a trend may be shifted due to bias, understanding that the trend exists is still useful.

Critical, high impact errors should always be evaluated on a case-by-case basis.

Keep in mind that metrics only tell us information about issues or errors that are successfully detected. In any thorough evaluation of a process, we should consider that errors can still go undetected despite our best efforts. This will eventually happen regardless of the implementation of a risk-based strategy or not. The hope is that a risk-based approach can shift these undetected errors to low impact elements and focus programming and validation efforts to reduce errors in the areas of greatest impact and value.

CONCLUSION

Metrics are critical for defending any risk-based approach whether it be data monitoring or programming validation. For the power of metrics to be fully realized, information needs to be collected in a consistent (and preferably controlled) manner. Our primary aim is to relate metrics to quality control in a risk-based validation setting. Metrics can be used to evaluate the appropriateness of risk assessments and subsequent testing methodologies of different programming elements. However, they can also be leveraged to provide support for addressing other operational challenges such as training, documentation and data flow, all of which may be indirectly related to risk assessment.

If collected appropriately, metrics can further provide additional insights across iterations or projects, not just within a single deliverable. This may also be elucidate systematic trends. We acknowledge that metrics may not provide all of the answers, and do not suggest that they always identify causal relationships. However, they can be informative. It is better to plan ahead so that data are available for metrics rather than trying to retrospectively collect information or try to reconstruct why a failure happened.

At end of the day, never lose sight that we are human. Despite our best efforts, errors will occur. If metrics are available to evaluate your quality control efforts, you will have a better chance of minimizing these events. A well-monitored, well-designed risk-based approach can help ensure that effort is focused where it matters most and that if errors do slip through your process, they will have a diminished impact on your overall reporting effort.

REFERENCES

[1] Food and Drug Administration. Guidance for Industry: Oversight of Clinical Investigations – A Risk-based Approach to Monitoring. Accessed March 13, 2018. https://www.fda.gov/downloads/Drugs/GuidanceComplianceRegulatoryInformation/Guidances/UCM269919.pdf.

[2] Marrer, Jennier, Risk-based-Program-Validation-for-Clinical-Trial-Analysis-and-Reporting, PhUSE US Connect 2019

[3] Randall, A. and Coar, W. Risk-based Validation in Clinical Trial Reporting: Focus on What Matters Most, PharmaSUG 2018

[4] Coar, W. and Randall, A., Data Monitoring Committee Report Programming: Considering a Risk-Based Approach to Quality Control, PharmaSUG 2017

CONTACT INFORMATION

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

Amber Randall
SCHARP – Fred Hutchinson Cancer Research Center
arandall@scharp.org

William Coar
Axio Research, A Cytel Company
williamc@axioresearch.com