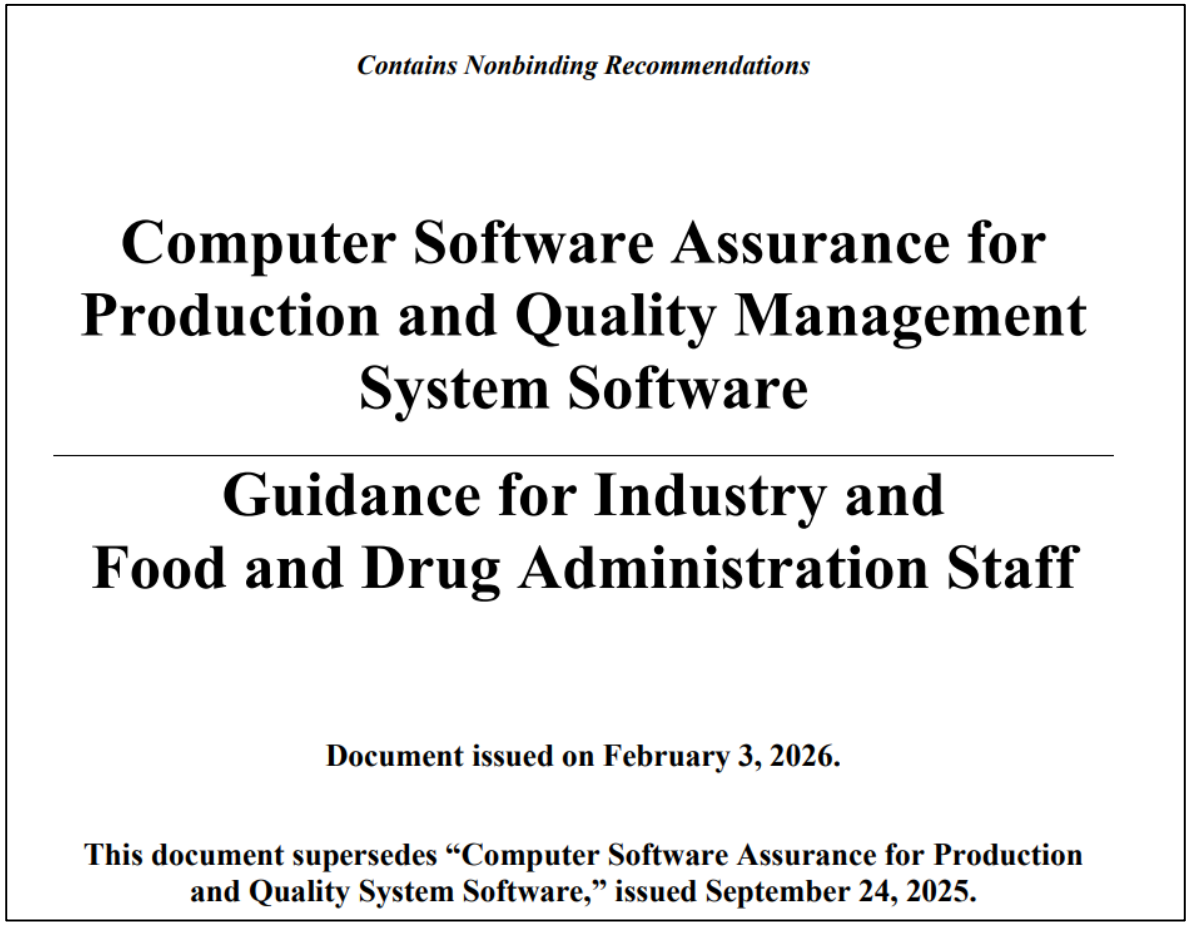
CSAとは？

CSA（Computer Software Assurance）とは？

- FDAのガイダンス「Computer Software Assurance for Production and Quality Management System Software」に記載された、ソフトウェアに対する品質保証の新しいアプローチのこと。

- ガイダンスには「医療機器」という用語が頻出しますが、製薬業界全体で取り入れてよいコンセプトだとされている。



Contains Nonbinding Recommendations

Computer Software Assurance for Production and Quality Management System Software

Guidance for Industry and Food and Drug Administration Staff

Document issued on February 3, 2026.

This document supersedes "Computer Software Assurance for Production and Quality System Software," issued September 24, 2025.

Computer Software Assurance for Production and Quality Management System Software | FDA

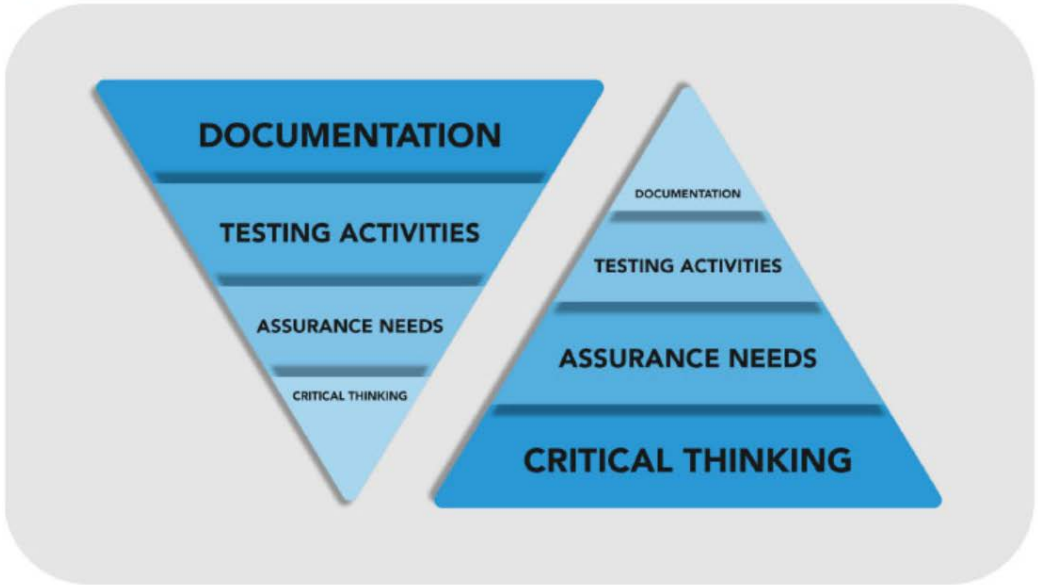
CSVと何が違うのか？

- 従来のCSVが、文書化重視、テスト重視だったのに対し、CSAはソフトウェアのリスクを評価して、リスクに応じた品質保証活動を行うとしている。

Table 1: Comparison of CSV to CSA (part 1).

CURRENT VALIDATION	CSA
Focus on creating documents/records for compliance and securing evidence for auditors (paper mountain effect)	Focuses on critical thinking and testing to achieve higher confidence in the quality of the system
Risk is based on failure mode effects analysis (FMEA) business risk and meeting regulatory requirement	Risk is based on the impact on patient safety, product quality (intended purpose), and data integrity
Validation involves testing all functionality (e.g., focus on COTS vs. potentially missing the higher-risk areas)	Validation is based on applying the right amount of diligence to a given level of risk to patient safety, product quality, and data integrity
Ignores previous assurance activity or related risk controls (e.g., one example is COTS functionalities)	Leveraging prior assurance activity, risk controls, and any documentation that exists with the vendor and others (e.g., build a good vendor risk management/vendor audit program that can be used to justify reduced COTS function testing)
All testing is scripted	Uses unscripted and ad-hoc testing in addition to scripted testing and employs the least burdensome approach to documentation
80% of time spent on documentation / 20% of time spent on testing	20% of time spent on documentation / 80% of time spent on testing

Figure 1: Comparison of CSV to CSA (part 2).



Computer Software Assurance and the Critical Thinking Approach | Pharmaceutical Engineering

ソフトウェアのリスクをどのように評価するのか？

- FDAのガイダンスの記載より

何等かの不具合が製品の安全性に関わるようあなソフトウェアはハイリスクだとしている。

Specifically, FDA considers a software feature, function, or operation to pose a high **process** risk **when its failure to perform as intended may result in a quality problem that foreseeably compromises safety, meaning a medical device risk**. This process risk identification step focuses only on the process, as opposed to the medical device risk posed to the patient or user. Examples of software features, functions, or operations that are generally **high process risk** are those that:

- Maintain process parameters (e.g., temperature, pressure, or humidity) that affect the physical properties of product or manufacturing processes that are identified as essential to device safety;
- Measure, inspect, analyze and/or determine acceptability of product or process with limited or no additional human awareness or review;
- Perform process corrections or adjustments of process parameters based on data monitoring or automated feedback from other process steps without additional human awareness or review;
- Produce instructions for use or other labeling provided to patients and users that are necessary for safe operation of the medical device; and/or
- Automate surveillance, trending, or tracking of data that the manufacturer identifies as essential to device safety (e.g., cybersecurity) and quality.

In contrast, FDA considers a software feature, function, or operation not to pose a high process risk **when its failure to perform as intended would not result in a quality problem that foreseeably compromises safety**. This includes situations **where failure to perform as intended would not result in a quality problem**, as well as situations **where failure to perform as intended may result in a quality problem that does not foreseeably lead to compromised safety**. Examples of software features, functions, or operations that generally are **not high process risk** include those that:

- Collect and record data from the process for monitoring and review purposes that do not have a direct impact on production or process performance;
- Are used as part the quality management system for Corrective and Preventive Actions (CAPA) routing, automated logging/tracking of complaints, automated change control management, or automated procedure management;
- Are intended to manage data (process, store, and/or organize data), automate an existing calculation, increase process monitoring, or provide alerts relevant to managing data when an exception occurs in an established process; and/or
- Are used to support production or the quality management system, as explained in Section V.A.1 above.

Computer Software Assurance for Production and Quality Management System Software | FDA

ソフトウェアのリスクをどうのように評価するのか？

Step1 そのソフトウェアは何のために使われているか？

区分	内容	例
製造・品質システムの一部として直接使用	規制対象プロセスに直接関与	製造設備制御ソフト、LIMS、電子バッチ記録
間接的な支援用途	直接関与しない補助的なツール	一般的なオフィスソフト、スケジュール管理ツール

Step2 プロセスリスク（Process Risk）の評価

- 「そのソフトウェアが意図通りに動かなかった場合、どれほどの影響があるか？」 を評価
- FDAは 「高いプロセスリスク（High Process Risk）」 と 「高くないプロセスリスク（Not High Process Risk）」 の2分類を採用

患者の安全や製品品質を損なう品質問題に関わるソフトウェアはハイリスクだとしている。

そのプロセスはHigh Riskか？

- 承認申請を目的として、臨床試験のデータ解析でRを使う
→High

- 臨床試験のデータ解析でRを使うが、試験途中の集計結果を社内に見せるだけ→Not High

- 臨床試験のデータ解析でRを使い、試験途中の集計結果を社内に見せ、その結果が何らかのアクションに繋がる→要検討

リスクに応じたテスト

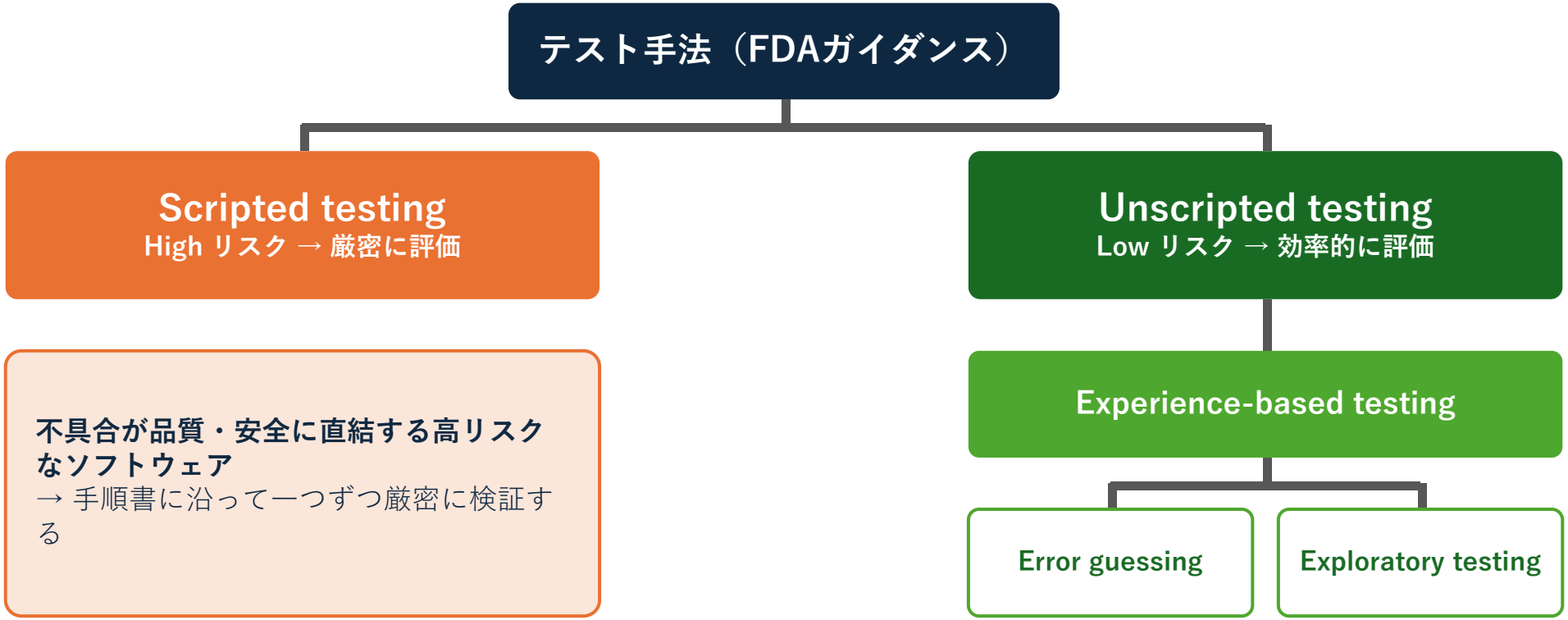
- FDAのガイダンスにはリスクに応じた以下のテスト手法が記載されている（一例）。
- Unscripted testing
  - Experience-based testing
    - Error guessing
    - Exploratory testing
- Scripted testing
- ソフトウェアのリスクに応じて、実施するテストも変えて良い。
  - Highリスク→Scripted testで厳密に評価する
  - Lowリスク→Unscripted testで効率的に評価

- **Unscripted testing**: Dynamic testing in which the tester's actions are not prescribed by written instructions in a test case.[19] It includes:
  - Scenario Testing (Also referred to as Ad-Hoc Testing): A specification-based test case design technique based on exercising sequences of interactions between the test item and other systems.[20] (Users are considered to be other systems in this context.)
  - **Experience-based testing**: Class of test case design techniques based on using the experience of testers to generate test cases.[21] Experience-based testing can include concepts such as test attacks, tours, and error taxonomies which target potential problems such as security, performance, and other quality areas,[22] and can include:
    - **Error guessing**: A test design technique in which test cases are derived on the basis of the tester's knowledge of past failures or general knowledge of failure modes. The relevant knowledge can be gained from personal experience, or can be encapsulated in, for example, a defects database or a "bug taxonomy."[23]
    - **Exploratory testing**: Experience-based testing in which the tester spontaneously designs and executes tests based on the tester's existing relevant knowledge, prior exploration of the test item (including results from previous tests), and heuristic "rules of thumb" regarding common software behaviors and types of failure. Exploratory testing looks for hidden properties, including hidden, unanticipated user behaviors, or accidental use situations that could interfere with other software properties being tested and could pose a risk of software failure.[24]
  - **Scripted testing**: Testing in which test cases are recorded (e.g., document in a test management tool or in a spreadsheet) and can then be executed manually or executed automatically using an automated testing tool. The level of detail required for each test case and the evidence necessary to establish the software feature, function, or operation performs as intended depends on the risk posed by the software feature, function, or operation. For example, depending on the intended use, a more robust scripted testing where the test cases and evidence may include detailed requirements for repeatability, traceability, or auditability may be appropriate.

Computer Software Assurance for Production and Quality Management System Software | FDA

リスクに応じたテスト

ソフトウェアのリスクの大きさに応じて、実施するテスト手法を使い分ける（FDAガイダンスより／一例）



出典：Computer Software Assurance for Production and Quality Management System Software | FDA

Scripted testing と Unscripted testing

Scripted testing（手順書化テスト）

- **定義：**テスト手順を文書化（手順書やスプレッドシート等に記録）し、手動／自動で実行する
- **記録する要素：**テスト目的／手順（ステップ）／期待結果／証拠／合否
- **特徴：**要件→手順→期待結果→実績の追跡性（トレーサビリティ）と再現性が高い
- **適用：高リスク（直接）機能 — 製品品質・患者安全に直結する機能**

Unscripted testing（非手順書テスト）

- **定義：**詳細なスクリプトによらず、担当者がその場で考えながら動的に実施する
- **手法：**探索的（Exploratory）／アドホック・シナリオ／エラー推測（Error guessing）
- **記録する要素：**テスト目的と合否はあるが、ステップごとの手順書は作らない
- **適用：高リスクでない（間接）機能 — 安全に直結しない機能**
- *注意：記録は必要。目的（チャーター）とテストログを残す — 「記録がなければ実施していないのと同じ」*

ポイント：リスクが高いほど厳密に（Scripted）、低ければ効率重視で（Unscripted）。ただし記録は必ず残す。

出典：Computer Software Assurance for Production and Quality Management System Software | FDA

日常の業務によるレバレッジ

- FDAのガイダンスによれば、日ごろから行っている業務でソフトウェアの品質保証を代替できるとされている。

(5) Additional Considerations for Assurance Activities

When deciding on the appropriate assurance activities, manufacturers should consider whether there are any additional controls or mechanisms in place throughout the quality management system that may decrease the impact of compromised safety and/or quality if failure of the software feature, function or operation were to occur. For example, as part of a comprehensive assurance approach, manufacturers can leverage the following to reduce the effort of additional assurance activities:

- Activities and established processes that provide control in production or fully verify processes in which software is involved. Such activities may include procedures to ensure integrity in the data supporting production, subsequent inspection or testing, or software quality assurance processes performed by other organizational units.
- Established purchasing control processes for selecting and monitoring software vendors. For example, the medical device manufacturer could incorporate the software development practices, validation work, and electronic information already performed by developers of the software as the starting point and determine what additional activities may be needed. For some lower-risk software features, functions, and operations, this may be all the assurance that is needed by the manufacturer.
- Additional process controls, including activities to reduce cybersecurity exposure,[26] that have been incorporated throughout production. For example, if a process is fully understood, all critical process parameters are monitored, and/or all outputs of a process undergo verification testing, these controls can serve as additional mechanisms to detect and correct the occurrence of quality problems that may occur if a software feature, function, or operation were to fail to perform as intended. In this example, the presence of these controls can be leveraged to reduce the effort of assurance activities appropriate for the software.
- The data and information periodically or continuously collected by the software for the purposes of monitoring or detecting issues and anomalies in the software after implementation of the software. The capability to monitor and detect performance issues or deviations and system errors may reduce the risk associated with a failure of the software to perform as intended and may be considered when deciding on assurance activities.
- The use of tools supporting software development and system life cycle activities (e.g., bug, anomaly tracking, requirement traceability tools) for the assurance of software used in production or as part of the quality management system whenever possible.
- The use of testing and results done in iterative cycles and continuously throughout the life cycle of the software used in production or as part of the quality management system.

レバレッジとしてガイダンスに書かれていること

- あくまで例示ですが、以下のような活動がレバレッジとして記載されています。
  - ソフトウェアが関与する工程を検証する既に確立されたプロセス
  - ソフトウェア購入時の一連の活動
  - ソフトウェアが異常な動作をしないか定期的・継続的に監視する活動
  - ソフトウェアのバグや異常を検知するためのサポートツール
  - ソフトウェアに対して継続的に実施されるテスト

CSAにおけるレバレッジ（活用）の考え方

レバレッジ＝適格なベンダー／サプライヤーが既に実施・文書化した活動を評価して活用し、重複する検証を避ける（最小限の負担で確信を得る）

① ベンダー評価
ベンダー／サプライヤーの品質システムの成熟度・実績を評価（監査・アセスメント）

② 活用候補の特定
活用できる証拠を洗い出す（ベンダーテスト・開発ライフサイクル統制・不具合管理）

③ 活用範囲の判断
自社のリスクと利用目的に応じて、どこまで頼れるかを決める

④ 不足分を補完
重複は省き、足りない確認だけ自社で実施（設定確認・利用目的に対するテスト）

重要：レバレッジしても最終責任は自社。ベンダーの成熟度を評価し、意図した使用に対する確認は必ず自社で行う。

Rにおいてはベンダー ＝ 開発者（コミュニティ）：
開発者（コミュニティ）のテストを活用し、利用者が必要最小限の負担で利用するプログラム（パッケージ）の品質を保証する。

出典：FDA「Computer Software Assurance for Production and Quality Management System Software」（一例。活用の可否はベンダーの成熟度とリスクに基づき判断）

レバレッジを実践するためのポイント

1 リスク評価を実施

対象システムの機能が「高リスク」か「低リスク」かを判断する

2 ベンダー監査（Supplier Assessment）

ベンダーが適切な品質管理のもとでテストを行っているかを確認する

Rパッケージの場合は、パッケージ開発者が実施したテスト記録の有無を確認

3 ベンダーのエビデンスを評価・収集

テスト結果・証明書・SOC2レポートなどを入手する

4 高リスク機能に追加テストを集中

低リスク機能はベンダーのエビデンスで代替する

レバレッジになりそうな業務（例）

- Rはフリーソフトであるためソフトウェア購入時の一連の活動はないが、通常行っている解析プログラミング業務にレバレッジになりそうな業務はないだろうか？

① ダブルプログラミング

② 実行ログの確認

③ コードレビュー

④ 他の解析結果との整合性の確認

⑤ 他のソフトウェアとの計算結果の比較

⑥ マニュアル・参考文献の参照

Rパッケージ開発者側の資料

- CRANの資料
- R validation hub

ディスカッション

- ディスカッション20分
- 発表5分

- Rパッケージのレバレッジとして考えられるものは、各パッケージで実施されたテスト記録の他にどのようなものが考えられるかディスカッションしてください。
- ディスカッションが終わったら、各テーブルから発表して頂きます。

Rパッケージと CSA

RパッケージとCSA

- Rを使って臨床試験のデータを解析し、その結果を承認申請に使うのであれば、R自体のCSVは必要であろう。

- しかし、追加でインストールして使う個々のRのパッケージはどうだろうか？CSAのコンセプトを使って、必要最小限のソフトウェアの品質保証活動にできないだろうか？

ディスカッション

- ディスカッション15分
- 発表5分

- 以下のRパッケージのリスクをHighとLowに分類して下さい。
- ディスカッションが終わったら、どのような結果になったか、各グループから発表して頂きます。

パッケージ名	説明	リスク
tidyverse	モダンなR解析には必須なパッケージ群	
survival	生存時間解析を行うパッケージ	
admiral	Pharmaverseが作成したADaM作成パッケージ	
sdtm.oak	Pharmaverseが作成したSDTM作成パッケージ	
xmpdf	PDFファイルのメタデータを操作するパッケージ	
labelled	変数ラベル、データフレームラベルを操作するパッケージ	
gtsummary	患者背景表や解析結果表を作成するパッケージ	
flextable	WordやPowerPoint向けの表を作成するパッケージ	
openxlsx	Excelファイルを操作するパッケージ（読込、出力、整形など）	

ディスカッション（40分）

- 個々のRパッケージに対してどのようなテストを行うかをディスカッションして下さい。その他に必要そうな品質保証活動があるか、ディスカッションして下さい。

- 個々のRパッケージに対して、日常の業務でレバレッジできる品質保証活動があるか、ディスカッションして下さい。

- 10分くらいで1つのパッケージのディスカッションを終え、その時点で、各グループからディスカッションの結果を発表して頂きます。

ディスカッションするRパッケージ一覧

グループ	パッケージ名	説明	リスク	実施するテストの内容	テスト以外に必要そうな品質保証活動	レバレッジになりそうな日常業務
1,2	tidyverse	モダンなR解析には必須なパッケージ群。				
3,4	survival	生存時間解析を行うパッケージ				
1,2	admiral	Pharmaverseが作成したADaM作成パッケージ				
3,4	sdtm.oak	Pharmaverseが作成したSDTM作成パッケージ				
1,2	xmpdf	PDFファイルのメタデータを操作するパッケージ				
3,4	labelled	変数ラベル、データフレームラベルを操作するパッケージ				
1,2	gtsummary	患者背景表や解析結果表を作成するパッケージ				
3,4	flextable	WordやPowerPoint向けの表を作成するパッケージ				
予備	openxlsx	Excelファイルを操作するパッケージ（読込、出力、整形など）				

終わりに

ソフトウェアの品質保証に関する臨床開発部門側の役割

- CSV専門の部署がある会社ばかりではなく、専門の部署がやるという決まりがあるわけでもありません。特にRのように、オープンソースかつフリーのソフトウェアを承認申請に使うという例は多くなく、臨床開発でデータ解析を担当するメンバーが果たす役割も大きいと考えられます。

1. CSV（全般）

1-1	ベンダーオーディット、CSV、システム導入を担当する部署が決まっていますか。

（事例・考え方）

専門部署があるケースや実施部門が行うケースいずれもあると考えられます。いずれの場合でも当該システムに対して責任を有する部署が検証していることが重要と考えます。

EI_20240321_eresqa.pdf

参考になるかもしれないDMの活動

- DMであれば、新しい臨床試験のたびにEDCのCSVを行っているので、CSVに関する一連の教訓は教えてもらえるかもしれません。

4. 教育啓発

4.1 デシジョンツリーおよび CSV ドキュメトセット表の教育啓発の仕方

現在、当局の適合性調査時の CSV に関する確認対象は EDC のみとなっており、ゆえに他のシステムに対する CSV 活動はおろそかになりがちである。また、我々は各社において、CSV 活動の教育を体系的に実施してきた経験はあるが、それではなかなか身につかず、実際に CSV 活動を始める段階で担当者が戸惑うことを多く目にしてきた。CSV の教育は、CSV を実際に担当する者を対象に、CSV 活動を始めるタイミングで教育することが最も効果的だと考えた。

gcp_system_validation.pdf

実は似たもの同士のRとSAS？

- オープンソース・フリーソフトウェアとプロプライエタリ・商用ソフトウェアという違いはあるものの、プログラムを書いて動かす、ログが自動的に残らないといった点でRとSASは共通点も多いと考えられます。
- SASで培った経験をRに活かすこともできるでしょう。

7-9　SAS/Excel 等の監査証跡を有さないツールでは、どのように ER/ES 指針の要件を満たせばよいでしょうか。例えば、入出力フォルダのバックアップ取得、アクセスコントロール（削除不可）や、入出力ファイルの保存等を行えばよいでしょうか。

（事例・考え方）

SAS/Excel 等の監査証跡を有さないツールやこれらのツールから出力されたファイルのみでは ER/ES 指針の要件を満たすことは難しいと考えられます。ご提示の方法を含む手順書の作成やチェックリストを利用した点検などにより、データ及びプロセスの信頼性を確保することが望ましいと考えます。

EI_20240321_eresqa.pdf

7.5　11.10(e) Use of secure, computer-generated, time-stamped audit trails to independently record the data and time of operator entries and actions that create, modify, or delete electronic records. Record changes shall not obscure previously recorded information. Such audit trail documentation shall be retained for a period at least as long as that required for the subject electronic records and shall be available for agency review and copying

The R Foundation understands this item to mean that the creation, modification or deletion of *records* must have an associated audit trail describing who, when and why an action was performed. Additionally, any such audit trail will be also considered an electronic *record* within the scope of Part 11.

R is not intended to create, maintain, modify or delete Part 11 relevant *records* but to perform calculations and draw graphics.

Where R's use may be interpreted as creating *records*, however, R can support audit trail creation within the *record*.

R includes `date()`, `Sys.time()`, `Sys.Date()` and `Sys.timezone()` functions which enable users to include date and time stamps on report, graphical and other output, thus enabling the use of this information in the tracking of user sessions.

R-FDA.pdf

Rにどう向き合うべきか

Rにどう向き合うべきか

- 統計処理とは，本来極めて能動的な営みである．どの手法を用い，いかなる論理を組み立てるか．その一つひとつの選択に解析者の主体性が宿らねばならない．そこに「無料」や「フリー」という記号ばかりが独り歩きしていることに，私は強い違和感を覚えるのだ．
- 私たちが真に求めるべきは，コストの不在ではなく，自らの手で解析を完結させ得る「能動性」であり，何ら制限なくそれを伝える術である．内部の機序を問わず，ただ与えられた機能を消費するだけの姿勢を続けるならば，早晩，その統計処理の座はAIに取って代わられるだろう．
- 統計処理の主権をAIに明け渡さぬためには，単なる「利用者」であることを辞め，能動的な「探求者」であらねばならない．自立した意思によって道具を飼い慣らす．それこそが，本来あるべき姿ではないだろうか．

K207.pdf