# Automated Validation Report

aNCA v0.0.0.9095

F. Hoffmann-La Roche Ltd

# Contents

# 1 Introduction

This document outlines the validation plan and results for R package "aNCA" version 0.0.0.9095. This document includes an overview of validation activities, a summary of the validation activity findings, and a report of the result of validation activities for the R package "aNCA" version 0.0.0.9095 generated using `autovalidate`.

## 1.1 Document Information

**System Name**: `aNCA` (0.0.0.9095)
**Document Name**: Automated Validation Report (`aNCA` (0.0.0.9095))
**Document Owner**: System Owner
**Document Location**: Approved Artifacts Stored in Veeva Quality Docs
**Associated Documents**:

- CSV (http://mycsv.roche.com)
- Autovalidate Validation Report
- System Risk Assessment
- R-Package Validation with Autovalidate Tool Operation Support Plan

Associated documents IDs are available in the Validation Registry.

## 1.2 Document History

| Version | Reason of Change | Effective from |
|---|---|---|
| 01 | First Release | Date of Approval |

### 1.2.1 Document History Version Notes

#### 1.2.1.1 1.2.1

- Update sub-sections ordering in traceability matrix to account for off-set introduced by the new Vulnerabilities section.

#### 1.2.1.2 1.2.0

- Add sections related to the vulnerabilities scanning (`Open Source Vulnerabilities Database` and `Vulnerabilities scan results`).
- Rephrase paragraph in the `Intended Use` clarifying `autovalidate` validation.
- Update Autovalidate SRA in the `Validation Registry`.

#### 1.2.1.3 1.1.0

- Remove skipped tests from the Testing Summary to avoid possible misinterpretations.

#### 1.2.1.4 1.0.9

- Updated System Risk Assessment entry in the Validation Registry.

#### 1.2.1.5 1.0.8

- Replace `revdepcheck` with `checked` as reverse dependecy check tool.

#### 1.2.1.6 1.0.7

- Updated System Risk Assessment entry in the Validation Registry.

#### 1.2.1.7 1.0.6

- Rephrased some parts of the Validation Plan to reduce potential ambiguities.

#### 1.2.1.8 1.0.5

- Introduce a hook for a traceability appendix to be injected, allowing for further characterization of project-specific requirements. (!81 @kelkhofd)
- Added **PD ALM IT QA** role.

#### 1.2.1.9 1.0.4

- Updates the "Responsible Persons" section to avoid amibuity about who assumes responsibility for external packages. (!80 @kelkhofd)
- Updates the "R Developers & Maintainers" entry in the Validation Plan to more accurately describe the responsibilities (or lack thereof) attributed to general package authors, and make it more clear how internal package authors may be involved in the process (!77 @kelkhofd)
- Removes duplicate extra "System Description" entry in the Validation Plan (!76 @kelkhofd)

#### 1.2.1.10 1.0.3

- Add Unknown status to the *Testing Summary* table and the rationale when such status is possible to avoid possible misinterpretations of the traceability matrix results. (@maksymis)

#### 1.2.1.11 1.0.2

- Remove cran-comments section to reflect changes in validation philosophy (@maksymis)
- Update documents IDs in Validation Registry (@maksymis)

#### 1.2.1.12 1.0.1

- Describe sampling method in the revdepcheck section to reflect overall stratefy changes implemented in autovalidateutils. Also mention usage of internal `revdepcheck` copy (@maksymis)

- Add testing session info to the `Test Execution Information` section to accurately present testing proccess. (@maksymis)

#### 1.2.1.13 1.0.0

- Updated `SRA` document ID from SRA-10494 to SRA-17954 (CHG0840298) (!67, @kelkhofd)
- The documented storage location changed from *Project Library* to *Veeva Quality Docs* (!66, @kelkhofd)
- *Responsible Parties* section revised to make it clear that the package authors are not responsible parties in cases where they are external to Roche. In which case the *Autovalidate R Support Team* takes responsibility for the validity of external software.
- Separated the *Validation Lead* role into the *Autovalidate R Validation Lead* and *Target System Validation Lead* roles to make the handoff of software for other systems more clearly defined.

#### 1.2.1.14 0.1.0

- Adding *Document History Version Notes* section where template changelog can be embedded as part of the report.
- `test.Rout` and `NEWS` contents are now wrapped to a new line after an 80-character limit to prevent overflow off the page when rendered to pdf (#90, @kelkhofd).

# 2 Software Overview

## 2.1 R Package Summary

Package `aNCA` (0.0.0.9095)

*"(Pre-)Clinical NCA in a Dynamic Shiny App"*

This application enables users to upload their datasets and perform Non-Compartment Analysis (NCA) on both pre-clinical and clinical datasets, with the results being easily visualizable. The NCA can be tailored to calculate pharmacokinetic parameters for various dosing regimens and time points, given certain restrictions. It also features manual slope selection, simplifying the process of conducting lambda-z-regression and PK-timepoint exclusions. Furthermore, the pharmacokinetic parameters can be dynamically visualized through customized graphics such as line and mean plots. The calculated pharmacokinetic parameters can be compiled in a dynamic table, visualized using boxplots, or exported as a comprehensive report. Designed with user-friendliness in mind, this app aims to make NCA accessible and straightforward for all scientists.

## 2.2 Responsible Persons

The provided R package is authored by Ercan Suekuer `ercan.suekuer@roche.com` [aut, cre] (`https://orcid.org/0009-0001-1626-`
Gerardo Jose Rodriguez `gerardo.jrac@gmail.com` [aut] (`https://orcid.org/0000-0003-1413-0060`), Pascal Baertschi `paesce.26@gmail.com` [aut] (`https://orcid.org/0000-0002-6533-0399`), Jana Spinner `spinnerjana@gmail.com` [aut] (`https://orcid.org/0009-0009-2197-9530`), F. Hoffmann-La Roche AG [cph, fnd], and maintained by Ercan Suekuer `ercan.suekuer@roche.com` [aut, cre] (`https://orcid.org/0009-0001-1626-1526`).

> Author roles are characterized using the roles notation specified in section *The DESCRIPTION file* of "Writing R Extensions". Regardless of authorship, the *Autovalidate R* support team assumes responsibility for package.

# 3 Validation Strategy

## 3.1 Validation Plan

### 3.1.1 Purpose

The purpose of the Validation Plan section is to

- outline the activities and deliverables for validation and testing of an R-package for subsequent deployment to a validated environment, where end users may use the R-package to support the delivery of content and code to Health Authorities (HA).
- outline the roles and responsibilities during the course of the validation and testing activities and for the generation, review and approval of the deliverables.

This section is developed in accordance with the governing Roche Computerized System Validation Policy (pol000014) and Computerized System Validation Lifecycle Directive (dir000202). The deliverables and activities discussed in this plan will provide the documented evidence that the R package performs according to its intended use, and has been sufficiently tested and deemed validated.

### 3.1.2 Intended Use

The decision to validate R packages with an in-house process using the Autovalidate Tool is documented in the Software Validation using Autovalidation System Risk Assessment (SRA-22510). The validation of the Autovalidate Tool was completed by the DIA group as referenced in Autovalidate Validation Report (SAP ID: 20271310). This Validation Plan has been established for the specific intended use of the autovalidate tool within the internal R package validation process.

An R package is a collection of R functions, compiled code and data. Users may use it in support of producing content and code for submission to Health Authorities. This may include, but is not limited to, the creation of figures such as tables, listings or graphs to support a submission or filing with a Health Authority. R packages may be used to produce and facilitate the generation of regulatory deliverables, and to integrate with internal systems to support this development. For reproducibility, R packages should be uniquely identifiable and documented, preferably using a uniquely identifying hash of the required source code files. Commonly, such a hash can be used within a version control system, such as `git`, in which case the hash used by such a system is preferred.

The validation process is supported by the `autovalidate` utility, which serves to provide validation checks in a continuous code development cycle. The `autovalidate` utility can be used to arrange R package metadata within a template which includes the validation strategy and report to produce a validation strategy and report document. The resulting document is in the form of a markdown-formatted plain-text file, which may be rendered into a more reader-friendly format such as a PDF. This document may then be stored in an approved EDMS. The `autovalidate` pipeline may be executed by an automatic process, reactive to any changes in an R package's code base, commonly referred to as a continuous integration process. The primary users of such a process will be internal R package developers who wish to integrate the validation of their R package with their development cycle. Quality, testing and validation teams may also use this approach to ensure that the process is providing the output for the intended use and purpose meeting requirements, as expected.

### 3.1.3 Validation Scope

#### 3.1.3.1 System Overview

This validation applies to R packages as extensions to systems on which the R programming language is installed for use in their support of analysis for regulatory submissions. The R language is an open source project with the source code freely distributed. This validation approach extends to R packages whose source code available. Given the source code of an R package, the software can be verified, compiled and installed by using built-in R language features for assessing its conformance to expected structure, best practices and installation.

R packages provide general extensions of the R language, providing function libraries, additional data structures, and syntactic conveniences. Frequently, these provide functionality for the creation of graphics, analytic outputs and data handling operations which might be used to support the creation of tables, listings and graphs for submission to health authorities.

If a package is found to be of sufficient quality as described in Section *Validation Approach*, the package is recognized as validated. A validated package may be installed into a computing system following a process specific to the target system. A conservative installation approach may prefer to re-validate all packages following the update of any package on the target system. Alternatively, a risk-tolerant deployment might be satisfied with the individual package validation report derived on a system that appropriately approximates the system into which the package is to be installed.

#### 3.1.3.2 In Scope

The scope of this validation plan will focus on the R package mentioned in the introduction section. The R package will be assessed and tested by Roche `autovalidate` utility, used for documenting this validation and testing activities. If necessary, subject matter experts will review these findings to provide further clarification and remediation. For the purpose of the development work and validation, anonymized and randomly generated data will be used to ensure GDPR compliance.

The testing includes high-level requirements which must be satisfied by all R packages, as well as R package-specific testing which covers the user-facing functionality of the specific R package. The high level requirements cover the unique identification of the source code, the installation of the R package into the target environment and adherence to general R package development best practices.

#### 3.1.3.3 Out of Scope

The following processes, functionalities and topics are out of scope for this validation plan:

- This validation process begins with the package source code. Software development and version control tools are not covered.
- The deployment of a package into a deployment environment, including supporting processes regarding disaster recovery, backup, supply and administration of package libraries and security.
- This validation process covers only an individual R package's adherence to a set of development standards in isolation, relying on the R language's support for facilitating the interactions of R packages. Further integration guarantees would need to be covered by a separate validation process.
- Explicit user acceptance testing is out of scope. Instead, this process assumes a close relationship between R package developers and users, such that any user acceptance concerns are communicated to the R package developers.
- Any operational assessment of risk associated with health authority submissions is strictly out of scope.
- The deployment of automated processes to execute the `autovalidate` and `autovalidateutils` tools in support of the validation process.

### 3.1.4 Roles and Responsibilities

The roles and responsibilities needed for perform this validation are described below:

**R Developers & Maintainers**:

May be either be employees of Roche or external to Roche. Neither internal nor external package authors are responsible for the validated state of the package. Though authors internal to Roche may be contacted to help determine the risk posed by a package, they are not held ultimately responsible for maintaining the validated state of the software.

The behavior and history of the authors may be used as evidence for justifying a risk assessment, including the authors':

- Providing resources to deliver, support, maintain and retire software applications throughout the lifecycle of the computerised system.
- Participating in the issue management process (as needed).

- Ensuring, if applicable, the custom/bespoke software code is reviewed according to established standards and procedures and is documented.
- Ensuring that software development is performed according to internal and/or external documented standards, which are appropriate to the technologies being utilized to develop the software.

The extent to which such practices are followed may be reviewed by the **Autovalidate R Validation Lead** to inform a risk assessment when other indicators of quality draw the risk posed by the package into question.

**Testing Experts** are responsible for:

- Providing input into the quality of the testing strategy
- Providing input into the developer-assessed risk of an R package, helping to revise a developer assessment if needed
- Operating as a supplementary R package developer by contributing risk-mitigation code contributions such as additional tests or changes to the source code

A **Product Development, Approval and License Maintenance IT QA (PD ALM IT QA)** is responsible for:

- Providing an oversight to ensure that a computerized system meets regulatory requirements and complies with internal processes.
- Ensuring validation activities are planned, executed and finalized in compliance with CSV Policy, Directive and SOP.
- Advising as required on external regulations, and internal policies and procedures.

An **Autovalidate R Validation Lead** is responsible for:

- Determining the validation approach, identifying deliverables needed or impacted by the Autovalidate R project / enhancement / change for the system.
- Enabling the validated delivery of software development and operations releases.
- Reviewing risk assessment and control activities.
- Supporting project phase activities as defined in computerized system validation plans.
- Supporting lifecycle processes such as change control and document management.
- Supporting training related to computerized systems quality and compliance, risk assessment and use of the supporting tools `autovalidate` and `autovalidateutils` in support of report authoring.
- Ensuring that all validation related document deliverables are being created and all appropriate tasks for system validation are executed.
- Providing support to IT operations teams by assessing the impact of changes on the validated state of the system.
- Reviewing defects and deviations during testing to ensure they are documented and closed.
- Accountable for CSV deliverables and activities.

A **Target System(s) Validation Lead** is responsible for:

- Verify the validation activities are completed by reviewing/approving the the validation report
- Ensuring that any open defects and deviations are documented and closed.
- Validation activities required in the Target System, if any.
- The overall validated state of the Target System.

A **Testing Lead** or **Testing Manager** is responsible for:

- Reviewing the `autovalidate` validation report creation process
- Periodic review of resulting validation report documents to ensure that the documented measures of R package quality appropriately reflect the level of software quality

A **Business Authority** is responsible for:

- Approving the `autovalidate` validation report creation process, and thereby the R packages which are successfully validated

A **Process Owner** is responsible for:

- Ultimately responsible for the validated state of the system.
- Approving key documentation as defined by plans and procedures.
- Provide adequate resources (personnel including SMEs, and financial resources) for system development and operation.
- Ensuring adequate training for end users.
- Ensuring that procedures required for systems operation exist, are followed, and periodically reviewed.
- Reviewing assessment/audit reports, responding to findings, and taking appropriate actions to ensure business processes are in compliance with GxP requirements.

A **Delivery Service Manager** is responsible for:

- Accountable for lifecycle management, performance, continuous improvement and overall customer satisfaction of a solution.
- Approval of deliverables
- Supporting the use of `autovalidate` and `autovalidateutils` for the production of validation reports
- Supporting business use of these tools
- Participating in issue management, especially as it pertains to system integrations which may include continuously integrated validation processes (as needed).

### 3.1.5 Validation Approach

According to System Risk Assessment, the R package must be validated. After validating the R package it will be allowed to be used for regulated business processes. The aim of validation activities is to show that the R package meets the specified intended use and can be maintained in the validated state.

Validation of the R package using `autovalidate` tool will include:

1. Extraction of R package metadata using `autovalidateutils`.
2. Successful execution of R programming best practices using the `R CMD check` utility distributed with the R language.
3. Successful execution of the testing plan.
4. Successful evaluation of the outputs of the testing plan using `autovalidate`, ensuring a satisfactory level of R package quality.
5. Successful creation of a validation report.

The validation activities reflected in this plan are aligned with the standards defined in CSV and PMM in combination with a risk-based approach. The basis of validation is prospective validation.

The specific validation activities relate to:

- Creation of a Validation Plan (this section) and Validation Report (this document).
- Carrying out the activities identified in the Validation Plan.
- Collection of the required documentation and records into a Validation Registry (this report).
- Explanation of the baseline configuration and the rationale for validation.
- Description of the validation approach.
- Reference to the Operational Support Plan which will outline how the validated state of users' computing environments is planned to be maintained. The Operational Support Plan will include Support Strategy (including Training), Configuration Management, Service Time, Functional and System Monitoring, System Administration, Archiving, Backup and Restore, Event Management, Incident and Service Request Management, Problem Management, Change Management, Service Level Management, Availability Management, Capacity Management, Continuity and Disaster Recovery, Security Management, Management of User Accounts and User access rights, Interface Management, Document Management, Periodic Audits/Reviews.

Validation activities will be driven by Computerized Systems Validation Policy [pol000014].

1. As part of the Validation Report generated by `autovalidate`, the following deliverables applicable for the `aNCA` package will be provided:
    - **Validation Plan** (this section of Validation Report) describes how the validation will be performed
    - **User Requirements Specification** defines High Level Requirements from user perspective
    - **Solution Architecture/Design Specification** describes in detail how the solution is built and implemented
    - **Test Plan** describes testing strategy approach, test environment
    - **Functional Requirements Specification** contains unique requirements for the target R package
    - **System Description** describes and specifies expected behaviors that any R packages should provide
    - **Test Cases & Test Report** provides information on such topics as Release Notes, Installation Verification, Test Description, and Test Results
    - **Traceability Matrix** tracks the relationships between Functional Requirements Specification and test deliverables
    - **Validation Registry** identifies and references all valid and effective documents associated with the system
2. The Operational Support Plan records plans for the activities, roles, responsibilities, and procedures for operational support, post-validation and during the lifecycle of the R package validation process

### 3.1.6 Activity Sequence

The validation lifecycle of R packages consist of the following minimal phases: Software Validation and Retirement. While the validation plan ends after the initial software validation of the package source code, the use of this source code must be managed by systems which implement their own software lifecycle management. This limited scope is due to the narrow focus of this validation plan, pertaining specifically to one version of source code and one testing computing environment. The validated product is independent of a computing environment and it is at the discretion of systems administrators and end users to evaluate the applicability of the validation report for their use.

#### 3.1.6.1 Validation Activity

The detailed validation activities related to the validation of the Autovalidate R tool are described in sequence:

#### 3.1.6.2 Supplemental Processes

##### 3.1.6.2.1 Create System Risk Assessment

The System Risk Assessment has been completed to identify, document and assess the system's overall risk level. Depending on the System Risk Assessment Summary, further project related actions are triggered or system maintenance activities may be arranged; including, but not limited to, Validation. The System Risk Assessment was conducted in accordance with the GI Computerized System Risk Management SOP (sop031554).

**Prerequisites**: Not Applicable

**Deliverable**: System Risk Assessment

**Acceptance Criteria**: Approval of the System Risk Assessment document

##### 3.1.6.2.2 Write Data Classification Report

R packages within themselves, do not manage, store or process sensitive data. R packages will be deployed to a validated R environment, which is used to run code from the R packages that can create reporting outputs for processing of data files and generation of associated output files. Due to this fact, data classification should be considered within the deployment environment. At

the time of writing, such relevant Data Classification Reports may include that of enableR (DCR-813), as documented in the SRA. Future systems where validated R packages may be deployed will require their own Data Classification Report before this process can support their validated use.

**Prerequisites**: Approved System Risk Assessment

**Deliverable**: Data Classification Report of Target Environments

**Acceptance Criteria**: Approval of the Data Classification Report

#### 3.1.6.3 Supporting Processes

##### 3.1.6.3.1 IT Procedures and Controls

Identify and establish processes for Support Strategy (including Training), Configuration Management, Service Time, Functional and System Monitoring, System Administration, Archiving, Backup and Restore, Event Management, Incident and Service Request Management, Problem Management, Change Management, Service Level Management, Availability Management, Capacity Management, Continuity and Disaster Recovery, Security Management, Management of User Accounts and User access rights, Interface Management, Document Management, Periodic Audits/Reviews.

**Prerequisites**: Not Applicable

**Deliverable**: Standard Operating Procedures for: Support Strategy (including Training), Configuration Management, Service Time, Functional and System Monitoring, System Administration, Archiving, Backup and Restore, Event Management, Incident and Service Request Management, Problem Management, Change Management, Service Level Management, Availability Management, Capacity Management, Continuity and Disaster Recovery, Security Management, Management of User Accounts and User access rights, Interface Management, Document Management, Periodic Audits/Reviews.

**Acceptance Criteria**: Approval of Support Strategy (including Training), Configuration Management, Service Time, Functional and System Monitoring, System Administration, Archiving, Backup and Restore, Event Management, Incident and Service Request Management, Problem Management, Change Management, Service Level Management, Availability Management, Capacity Management, Continuity and Disaster Recovery, Security Management, Management of User Accounts and User access rights, Interface Management, Document Management, Periodic Audits/Reviews SOPs.

##### 3.1.6.3.2 Business Procedures and Controls

Identify and establish processes for End-User Operations, Business Continuity Management, Document Management, Training Management.

**Prerequisites**: Not Applicable

**Deliverable**: Standard Operating Procedures for End-User Operations, Business Continuity Management, Document Management, Training Management.

**Acceptance Criteria**: Approval of End-User Operations, Business Continuity Management, Document Management, Training Management SOPs.

#### 3.1.6.4 Business and IT Required Training

##### 3.1.6.4.1 Business and IT Training Requirements and Materials

Identify training requirements and create required training materials for R package developers, users of R packages and IT implementation and support personnel responsible for downstream systems considering the use of validated R packages.

**Prerequisites**: Not Applicable

**Deliverable**: R package developer, users of R packages and IT implementation and support personnel training materials.

**Acceptance Criteria**: Final Training Materials for R package developers, users of R packages and IT implementation and support personnel.

#### 3.1.6.4.2 Training of Software End-Users

Develop training materials for R package end-users.

**Prerequisites**: Not Applicable

**Deliverable**: Trainees signatures on training attendance sheet and/or personal training record.

**Acceptance Criteria**: Instructor's signature on training attendance sheet and/or personal training record.

#### 3.1.6.4.3 Training of Software Developers

Develop training materials for internal R package developers.

**Prerequisites**: Not Applicable

**Deliverable**: Trainees signatures on training attendance sheet and/or personal training record.

**Acceptance Criteria**: Instructor's signature on training attendance sheet and/or personal training record.

#### 3.1.6.4.4 Training of Personnel Involved in Testing Activities

Develop training materials and perform training for all personnel participating in or configuring testing activities. Appropriate training includes applicable documentation, testing procedures and applications.

**Prerequisites**: Not Applicable

**Deliverable**: Trainees signatures on training attendance sheet and/or personal training record.

**Acceptance Criteria**: Instructor's signature on training attendance sheet and/or personal training record.

#### 3.1.6.4.5 Training of IT Support Personnel

Train all system support personnel, whose responsibilities may include responding to user concerns of package installations or administrating the deployment of R packages. Such training must include considerations which inform the applicability of a validation report for the use on a deployment system. Specifically, this applies to assessing the similarity of the testing environment to the deployment environment for the purposes of validated R package use and steps which may be considered to confirm this assessment.

**Prerequisites**: Final Training Materials

**Deliverable**: Trainees signatures on training attendance sheet and/or personal training record.

**Acceptance Criteria**: Instructor's signature on training attendance sheet and/or personal training record.

### 3.1.6.5 Planning Stage

#### 3.1.6.5.1 Write Validation Plan

Creation of the Validation Plan for R package validation (this document section) defining how the validation of R package will be performed. This deliverable describes the activities and deliverables require to provide documented evidence for the validation of R package.

**Prerequisites**: Approved System Risk Assessment.

**Deliverable**: Validation Plan

**Acceptance Criteria**: Approval of the Validation Plan.

#### 3.1.6.6 Define Stage

##### 3.1.6.6.1 Develop Requirement Specification

Define and document the generalized expectations of R package.

**Prerequisites**: Not Applicable

**Deliverable**: The User Requirements Specification included within this document.

**Acceptance Criteria**: Approval of the User Requirements Specification.

##### 3.1.6.6.2 Develop Functional Specification Approach

Define which documented behaviors within R packages should be interpreted as functional requirements and document them.

**Prerequisites**: Requirements Specification

**Deliverable**: The method described and requirements documented in Functional Requirements Specification.

**Acceptance Criteria**: Approval of the method and requirements described in Functional Requirements Specification.

##### 3.1.6.6.3 Perform Functional Risk Assessment

Based on the SRA risk evaluation outputs (Business Related Risk as Medium and Technology Related Risk as Low) and according to CS Risk Management SOP031554, Functional Risk Assessment was deemed not required.

**Prerequisites**: Approved Requirements Specification and Functional Specification

**Deliverable**: Not Applicable

**Acceptance Criteria**: Not Applicable

##### 3.1.6.6.4 System Description

This deliverable includes an overview of R package as extensions of the R language, as well as a high level overview of the criteria which must be satisfied by any R package to be considered validated.

**Prerequisites**: Approved System Risk Assessment.

**Deliverable**: The System Description section of this document.

**Acceptance Criteria**: Approval of the System Description section of this document.

#### 3.1.6.7 Build Stage

##### 3.1.6.7.1 Develop Solution Architecture/Design Specification

Define and describe in detail how the solution is coded and/or configured, built and implemented.

**Prerequisites**: Approved Functional Specification

**Deliverable**: Solution Architecture/Design Specification

**Acceptance Criteria**: Approval of the Solution Architecture/Design Specification section of this document.

#### 3.1.6.8 Verification Stage

##### 3.1.6.8.1 Verify Test Environment

Verify that the R-package is installed following documented and controlled procedures.

**Prerequisites**: Qualified Test Environment

**Deliverable**: The Installation Verification section of this document.

**Acceptance Criteria**: Approval of the Installation Verification section of this document.

##### 3.1.6.8.2 Write Testing Plan

Define the scope of testing: how the testing is performed, how errors are handled, logging of failure modes and the criteria for software acceptance.

**Prerequisites**: Approved Validation Plan

**Deliverable**: The Testing Plan section of this document.

**Acceptance Criteria**: Approval of the Testing Plan section of this document.

##### 3.1.6.8.3 Verify Test Case Evaluation

Execution of the testing plan and documentation of execution logs and test findings.

**Prerequisites**: Approved Testing Plan

**Deliverable**: The `R CMD check` Results and Test Execution Log sections of this document.

**Acceptance Criteria**: Successful execution of `R CMD check` as described in `R CMD check` Results and successful test execution as evident by the lack of error messages emitted in the Test Execution Log.

##### 3.1.6.8.4 Verify Traceability Matrix Evaluation

Track the relationship between documented package objects, interpreted as a user contract of behavior requirements, and the tests which prompt their evaluation during the execution of the testing plan.

**Prerequisites**: Approved User Requirements Specification, Functional Requirements Specification, System Description and Testing Plan.

**Deliverable**: Documentation of any traceability matrix gaps in Autovalidate Execution Log

**Acceptance Criteria**: Successful requirement coverage as evident by the lack of warnings in the Autovalidate Execution Log section of this document.

##### 3.1.6.8.5 Write Validation Report

A summary of the validation carried out against the Validation Plan. The Validation Report will be produced detailing results of executing the Validation Plan. This documents all functional requirements presented by the package in the form of documented objects as well as the results of the execution of the Testing Plan to assess the package.

**Prerequisites**: Validation Report Template (the precursor template to this document), Approval of Test Case Evaluation, Approval of Traceability Matrix Evaluation.

**Deliverable**: The Validation Report (this document).

**Acceptance Criteria**: Approval of the Validation Report Template, Successful Test Case Evaluation, Successful Traceability Matrix Evaluation.

#### 3.1.6.8.6 Complete Validation Registry

This deliverable identifies and references all valid and effective documents associated with the target R package validation.

**Prerequisites**: Approved Validation Report

**Deliverable**: Validation Registry

**Acceptance Criteria**: Approval of Validation Registry

### 3.1.7 Validation Acceptance Criteria

The fulfillment of the following acceptance criteria will constitute an acceptable execution of this plan:

1. The R package has been successfully tested, according to the Test Cases & Test Report section, and deemed validated upon internal acceptance:
    - The R package have been shown to be deemed fit for its intended use, as documented in Test Cases & Test Report section per execution and acceptance of testing results.
    - The validated R package is suitable as per acceptable standards for deployment to the validated target environment.
2. Package documentation is current and adequately describe how the R package will be operated, as assessed using the `R CMD check` utility.
3. Any exceptions from this plan are justified, open issues are documented and assessed, if applicable.
4. Required training has been completed and documented.
5. Validation documentation (SRA; Validation Report - inclusive of the Validation Plan, Validation Report and Validation Registry; Operational Support Plan) is reviewed and approved.

## 3.2 Terms and Abbreviations

The following terms will be used throughout this report.

**Computing Environment**: A computing environment describes the breadth of tools installed on a system which affect a computational process. In the context of R, this includes, but is not limited to hardware characteristics such as the processor architecture, system software such as the operating system, available system utilities, dynamic libraries and executables, and configuration settings such as environment variables. Throughout the lifecycle of an R package's development and deployment, the environment is relevant computing environment is likely to change. There are three classifications of computing environments, each relevant to a different part of the R package development and deployment lifecycle:

- **Development Environment**: A development environment is any computing environment in which changes to the source code are introduced.

    A developer performs their own testing as an approximation of how the source code might perform within the testing environment. It is up to the developer to decide what level of stringency they would like to impose for the types of contribution they were making, but ultimately these tests are only a convenience for approximating execution within the testing environment which is considered the point of truth.

- **Testing Environment**: A testing environment is a computing environment in which tests associated with the source code are evaluated.

  A testing environment reflects the target environment as closely as possible, while balancing the speed of execution against the reliability of results. The testing environment executes the testing suite of the software and is considered a point of truth.

  To illustrate considerations which might inform whether the testing environment adequately reflects the target environment, one might consider that the underlying low-level system libraries which are used by the system, but do not affect the R language or packages. Tests would expect to perform identically, irrespective of these libraries. However, package dependencies or system libraries which are used by the R language may affect the result of tests. For validated use of packages on systems for GxP relevant use, such environment differences would disqualify the package for validated use. If the target system is intended for work which is low risk or fault-tolerant, such differences might be acceptable and should be a consideration when interpretting package validation results.

- **Target (or Deployment) Environment**: A target (or deployment) environment is the computing environment into which the source code is intended to be used by end users.

  With a sufficiently representative testing environment, no further testing is needed to show that software which has been valdiated within the testing environment will perform as expected within a target environment. Throughout the course of use, end users evaluate the acceptability of behavior, and are encouraged to provide feedback directly to R package developers.

**Continuous Deployment**: Continuous deployment (CD, or along with continuous integration, often abbreviated CI/CD), is the process of triggering a downstream process as a result of source code changes. Frequently, this pertains to deployment of a new version of a service that makes use of the source code. For the purpose of R package validation, this might include registration of a new, validated version of an R package with an R package repository or archival of a validation report with an external document repository.

**Continuous Integration**: Continuous integration (CI) represents a class of automated computational tasks which execute in response to source code changes. Frequently, continuous integration is used to enforce quality constraints on code, including proper building of software, execution of all unit tests and constraints on how code coverage might change over time. For the purposes of validation, successfully achieving validation criteria.

**End Users**: The end users of an R package encompasses any consumer of the source code or its derivatives for purposes that do not contribute to its development. Most frequently, this includes users who are using the software for its explicit purpose to add functionality to the R language. In addition, this includes users who use the software in an administrative context, managing the installation or deployment of the software as a utility.

**Hash**: Source code is often uniquely identified by an automatically generated, uniquely identifying tag known as a hash. Using the `git` version control system, the SHA-1 method of generating a hash is used. This is a robust, established and widely used cryptographic function for generating hashes. Regenerating a hash allows developers to independently verify that source code mirrors source code used elsewhere. For the purposes of validation, this hash allows end users to verify that the source that they are using matches the source code against which a validation report was produced.

**R Package**: An R package is defined as a modular set of R language functionality. It is defined by source code which fully specifies the R package's capabilities and dependencies. For further details, see the *Packages* section of "An Introduction to R". Notably, though R packages offer facilities for distributing data, it is considered best practice to only include minimal, non-sensitive data for the purpose of example.

**R Package Contributors**: In source code development, "contributors" includes any individual who has played a role - in any capacity - which affects the content of the source code. Where "developer" assumes a level of familiarity with the source code, a contributor does not assume depth of knowledge about the project.

**R Package Developers**: In source code development, "developers" includes any individual who plays an active role in the writing or improving source code. This may include individuals either directly or peripherally affiliated with the project.

**R Package Library**: A collection of R packages installed within a directory within a computational environment.

**R Package Maintainers**: Source code maintainers, and specifically R package maintainers, include developers or administrators of an R package with a focus on the stability, maintenance and consistent quality of the source code. The role of

the maintainer offers a reliable and accessible point of contact for addressing source code improvements, maintenance and concerns.

**R Package Repository**: A collection of R package source code, archives or binaries for distribution. The R Project, which oversees the development of th R language, maintains the public R Package Repository CRAN. In addition, there are services and tools for the administration of private or alternative R package repositories. The R language offers built-in facilities for downloading and installing R packages from such repositories.

**Source Code**: The files that comprise an R package. In addition to the files required to build an R package, it's common for additional development-focused or user-focused files to be included in the source code. This may include unit tests, configuration files, additional documentation, additional files pertaining to the administration of the source code.

The following abbreviations will be used throughout this report.

| Abbreviation | Meaning |
|---|---|
| **CSV** | Computer System Validation |
| **DCR** | Data Classification Report |
| **EDMS** | Electronic Document Management System (Veeva Quality Docs) |
| **GDPR** | General Data Protection Regulation |
| **GI** | Group Informatics |
| **GxP** | Good *X* Practice (FDA Compliance; where *X* may be Clinical, Laboratory, Manufacturing, Pharmaceutical, etc) |
| **HA** | Health Authority |
| **IDM** | Integrated Document Management |
| **PMM** | Project Management Methodology |
| **QD** | Veeva Quality Docs |
| **QD ID** | Veeva Quality Docs Identifier |
| **SME** | Subject Matter Expert |
| **SOP** | Standard Operating Procedures |
| **SRA** | System Risk Assessment |
| **TLGs** | Tables, Lists and Graphs |
| **UAT** | User Acceptance Testing |

## 3.3 User Requirements Specification

The high level requirements outline expected behaviors of all validated R packages. These requirements ensure fundamental aspects of code quality and development practices and confirm basic capabilities of the R package as it interfaces with the R language.

### 3.3.1 Uniquely Identifiable Source Code

The source code used to build the R package must be uniquely identifiable, providing or permitting the calculation of a recoverable identifier which can be used to confirm the contents of the source code.

The ability to reproducibly identify the source code is captured and documented in *Section "Software Version Control Summary"*.

### 3.3.2 R Language Guideline Adherence

The R package should conform to expectations set by the R language, as documented in "Writing R Extensions". These checks can be automatically executed using utilities built in to the R language. Specifically, this pertains to the use of the `R CMD check` command line utility. With these checks passing, the R package can be reasonably expected to build and install successfully in a sufficiently similar deployment environment.

Among these checks, this utility will ensure that the R package can be successfully installed and loaded, providing sufficient confidence that the R package interfaces appropriately with the R language and other R packages.

The output of these checks are captured in *Section "`R CMD check` Results"*.

### 3.3.3 R Package Behaviors

In addition to general behaviors of all R packages, as assessed in *Section "R Language Guidance Adherence"*, the target R package provides the following high level description:

> `aNCA` *v0.0.0.9095*
>
> **(Pre-)Clinical NCA in a Dynamic Shiny App**
>
> This application enables users to upload their datasets and perform Non-Compartment Analysis (NCA) on both pre-clinical and clinical datasets, with the results being easily visualizable. The NCA can be tailored to calculate pharmacokinetic parameters for various dosing regimens and time points, given certain restrictions. It also features manual slope selection, simplifying the process of conducting lambda-z-regression and PK-timepoint exclusions. Furthermore, the pharmacokinetic parameters can be dynamically visualized through customized graphics such as line and mean plots. The calculated pharmacokinetic parameters can be compiled in a dynamic table, visualized using boxplots, or exported as a comprehensive report. Designed with user-friendliness in mind, this app aims to make NCA accessible and straightforward for all scientists.

The broad behaviors outlined here are provided through user-facing R package functionality which is described in *Section "Functional Requirements Specification"*, and tested as documented in the *Section "Traceability Matrix"*.

## 3.4 System Description

R packages provide additional functionality to the R language. They do this by packaging user-facing functionality, documentation and data in a standard file structure, which the R language can use to systematically integrate that functionality into an R session.

A validated R package satisfies quality criteria that assure to a reasonable level of confidence that the R package can deliver it's described functionality.

R packages allow developers and users

- to make user of a set of related functionality to supplement the base R language
- to easily share reusable code, such as common procedures, methodologies and repeated tasks
- to share reusable documentation and examples of how to use the provided code

> For a comprehensive overview of all R package behaviors, please refer to section *Packages* of "An Introduction to R".

Specifically, a validated R package fulfills the following:

- Ability to build and install according to the R language's built-in packaging building utilities
- Ability to build, install and load within the testing environment, providing sufficient evidence that the package can be installed within an environment that is suitably similar to the testing environment (most notably, including a similar operating system, relevant system libraries and that package dependency requirements have been met)

- Provides sufficient user-facing documentation of provided functionality and exported objects, which are interpreted as a form of requirements that describe a usage contract with the end user
- Sufficiently tests the documented user-facing behaviors, interpreted as software requirements

## 3.5 Solution Architecture / Design Specification

R packages provide a modular, composeable mechanism of distributing reusable R code. Validation of an R package is specific to a version of R package source code as installed within a specific computation environment.

### 3.5.1 R Package Architecture

An R package must conform to a specific directory structure, including software metadata including required fields such as the R package's name, version, description and itemization of R package dependencies, as well as optional metadata such as release notes and exported object documentation. The full structure of R packages is described in section *Package structure* of "Writing R Extensions".

#### 3.5.1.1 Package Installation

R packages are installed using R's built-in installation facilities, installing an R package from source or a gzipped package tar archives as specified in the *Installing packages* section of "R Installation and Administration". These files might be sourced from an R package repository, or built from source files prior to installation. If the R package source is validated for use in a validated system, all installation procedures provided through the base R language, or through the widely-used developer utilities R package, `devtools`, are valid mechanisms of installation.

Some examples of such installation procedures include, but are not limited to

1. Using `R CMD INSTALL` to install an R package from a source code directory of a validated version of the source code, or from a gzipped tar archive of a validated version of the source code.

   ```
   R CMD INSTALL "./aNCA"
   R CMD INSTALL "./aNCA.tar.gz"
   ```

2. Using `install.packages`, provided that the R option `"repos"` contains a path to an R package repository serving a validated version of the desired R package.

   ```
   install.packages("aNCA")
   ```

3. Using `devtools::install`, which is itself a shorthand for using `R CMD INSTALL`.

   ```
   devtools::install("./aNCA")
   ```

4. Using the `devtools::install_*` family of functions, which handle the download of source code from a variety of possible code repositories and subsequent execution of `devtools::install` to install the source code, provided that the target repository contains source code for a validated version of the R package.

   ```
   devtools::install_github("Organization/Repository")
   ```

#### 3.5.1.2 Package Dependencies, Constraints & Assumptions

The R package dependencies are managed by the R language, which has in-built functionality for ensuring that dependencies are satisfied. This includes any constraints on the R language versions which are applicable, as well as any additional R package dependencies required to run the R package. In addition, R packages denote exactly which R package dependencies are required to contribute to the R package's development. Installation accesses available R package repositories

for dependencies that are not already installed within the R environment. The scope of this validation process does not mandate a source for these dependencies, allowing for installation from a R package repository such as CRAN, an alternative R package repository, a source code archive, or from a source code repository. R packages available through these channels are assumed to be faithfully served R package source code or binaries, reflective of the R package behaviors. R packages are assumed to appropriately update their advertised R package versions when functionality changes, allowing for any R package dependency version constraints to effectively specify dependency behaviors.

The rigor and reproducibility of dependency installations for validating R packages should reflect the rigor with which the testing environment is constructed. To most closely match a development and testing environment, a centralized R package repository may be preferred to guarantee the exact reflection the R package and dependencies in the target environment. To be exhaustive, it is preferred that all dependencies have undergone validation prior to validationg a dependent package for regulatory use. However, it is permissible to incorporate ad-hoc `Suggests` dependencies. Such packages are often used for testing, and are not required for package use.

R packages required only for generating validation reports such as `autovalidateutils` follow the same validation procedure, but with the caveat that required dependencies which may have not yet been validated for regulated clinical use may be installed to support the process as a transient library. Because the core R utilities and processes of this validation pipeline rely on established processes (namely R CMD check, unit test evaluation and code coverage), the risk of errors being introduced within `autovalidateutils` that are note identified elsewhere in the R ecosystem is highly unlikely and does not warrant the extensive and duplicative effort needed to re-implement such processes independent of any dependencies.

These dependencies are itemized in *Section "Dependencies"* within the Validation Report.

#### 3.5.1.3 Package Version Control

For the purposes of this validation report, the version of the R package that is tested is defined by the version of source code required to build the R package. This is often uniquely identified by a hash.

The version control platform should provide a traceable record of the individuals that contributed code to the R package, a mechanism of reporting issues, and mechanisms of managing code changes and contributions. The adoption and use of these features is left to the discretion of the R package developers, who are expected to adhere to a level of stringency reflective of the perceived risk of code changes.

**R Package Development Environments**

The validated version of the R package is characterized by a specific version of its source code, verifiable using a uniquely identifying hash of the source code contents. The development of the R package source code is outside the scope of this validation report, permitting development of the source code to take place wherever the R package developers choose to develop their code. This may include, but is not limited to, an individual's personal computer, or a managed system. The development environment need not replicate the testing environment, though developers may find that it is easier to verify passing test cases if such an environment is used. The choice of development environment is left to the software developers, and is expected to take into consideration the sensitivity of the work and information required for development.

#### 3.5.1.4 Package Lifecycle Management

R packages installed through a managed repository, or with a remote source code repository are actively checked for R package updates and may be updated with the widely used `devtools` R package, which checks for updated versions or new contributions.

Individual R package functionality may also require lifecycle management to ensure users are aware of deprecations, experimental functionality or superseded user interfaces. These behaviors are best managed by the R package authors and should be communicated to appropriately reflect the level of risk of the change. Popular tools to support the communication of the R package development lifecycle might include the `lifecycle` R package or bespoke handling of user-facing communications.

### 3.5.2 Validation Execution Architecture

For use of a R package to be considered validated, the testing of the R package must take place within a computation environment reflective of the target system in which the R package is intended to be used. How such an environment is constructed is not considered to be in scope for this validation report. As examples, such an environment may include a bespoke computational platform upon which validated R packages are centrally managed, or it may include a base container image or virtual machine into which R packages are installed.

#### 3.5.2.1 Test Execution

Likewise, so long as the execution of the test suite occurs in such an environment, there are no restrictions on how such a validation report is produced. As examples, an individual may execute the testing suite, or it may be automatically executed as a continuously integrated facet of the R package development cycle.

#### 3.5.2.2 Validation Reporting

To report validation criteria, the `autovalidate` software is used to compile metadata and documentation to form a cohesive validation report. An R package validation report template is populated with R package metadata and diagnostic outputs to document infromation relevant to a validation decision.

There are a number of metadata files that provide R package metadata in a format which can be used by the `autovalidate` software. To produce these, the R package, `autovalidateutils` provides functionality which collects this metadata and formats it as expected by the `autovalidate` R package validation report template.

### 3.5.3 R Package Deployment Architecture

This validation report covers the validation of an R package as characterized by the version of the source code and its installation within the target environment, which the test environment is intended to reflect. This validation process is not intended to cover R package deployment, which may involve additional systems, software repositories and access controls.

In the simplest form, a user may acquire R package source code and use facilities provided by the R language and supporting tools (as described in Package Installation) to manage an installation themselves. To provide easier access to end users, a central repository of validated R packages could also serve these R packages, either as source code, R package archives or binaries pre-built in the target environment. Providing such a system would incur its own validation process, which is considered outside the scope of the validation of a single R package.

## 3.6 Testing Plan

R package testing represents a trail of provable adherence to requirements, as characterized in an R package's narrative text offerings including the R package description and documented objects. Testing of an R package is executed within a computational environment, executing a series of R language tools and utilities to characterize the coverage of described behaviors by the provided tests.

### 3.6.1 Test Environment

The testing environment describes the cohort of software tools available to the automated testing suite upon execution. In the most extreme characterization, this can include everything from the hardware upon which the system is evaluated, system operating system, available system libraries which might be necessary for execution, the R language itself and any necessary R package dependencies. In practice, the R language abstracts many of the nuances of these low-level systems to limit their impact on the evaluation of tests, and is for most uses sufficient in and of itself for approximating test reproducibility. To best approximate an environment into which the R package might be used, a container or virtual machine

that reflects the configuration of the target system is to be used to execute the R package's tests. Such an approach is sufficient for establishing confidence that the R package's tests will evaluate as reported once the R package is installed on the target system.

Details of the test environment that are relevant to the R language are detailed in the Installation System Information, where the output of diagnostic R language functions is provided, including the output of `sessionInfo`, `capabilities` and `libPaths` which collectively document the host operating system, available system libraries, loaded packages, R language compiled capabilities and library paths available to the system during testing. This information provides sufficient description of the testing environment, which may be used to decide the relevance of the test results for use on alternative systems.

**Package Interactions**

An environment into which validated R packages might be loaded may consist of cohorts of R packages, whose functionality may compose - either as designed explicitly by the R package authors, or through interactions introduced by an end user. The R language provides robust tools for articulating concerning interactions, which may include but are not limited to, dependency management, conflicts in a global namespace, masking of functions, required system utilities, management of lower-level language libraries and system utilities. The safeguards provided by the R language are deemed sufficient for communicating to the end user when provided functionality might be affected by the introduction of additional R packages.

### 3.6.2 Testing Tools

The R language is distributed with, or is extended by, a number of tools to evaluate R package best practices, tests and overall test efficacy. The following tools and resources are used for evaluating a package's adherence to these principles.

#### 3.6.2.1 Open Source Vulnerabilities Database

The R package and system dependencies required by it, are checked for known vulnerabilities using `Open Source Vulnerabilities` database. The `OSV` enables developers to identify risks posed by third-party open source dependencies and take necessary remediation steps. The `OSV` aggregates multiple different repositories which adopted `OpenSSF Vulnerability format` and provides infrastructure to ensure affected versions are accurately represented.

#### 3.6.2.2 R utility `R CMD check`

As a baseline indication of R package quality, the built-in R facility, `R CMD check` was used to evaluate the package. `R CMD check` issues a number of checks to evaluate an R package's structure, included documentation, and general development best practices.

#### 3.6.2.3 R Standard Library Function `tools::testInstalledPackage`

The most generic mechanism of evaluating R package unit tests is using the R standard library's `tools::testInstalledPackage` function, which evaluates code within the `tests` subdirectory of an R package. This function is used as the basis for more sophisticated testing strategies, such as for the default evaluation of code coverage using `covr::package_coverage`.

#### 3.6.2.4 R Unit Testing Packages

Commonly, the R package `testthat` is used to manage reproducible test evaluation. `testthat` provides a family of functions for evaluating sets of unit tests and provides the underlying functionality to `devtools::test`.

In the long history of the R language, predecessors to were used. Of these, `RUnit` is still in active use among some R packages, and others implement entirely bespoke testing strategies.

#### 3.6.2.5 R Package `covr`

In addition to the evaluation of unit tests, a common metric for evaluating R package robustness is code coverage. Code coverage is loosely defined, but most frequently refers to the percentage of lines or expressions within a body of code that are evaluated at least once by a unit test suite. Herein, code coverage will be defined specifically as the percentage of lines of code that are evaluated during a unit test suite execution.

`covr` is an R package which provides mechanisms of injecting traces into an R package during evaluation of unit tests to collect such metrics. These traces only exist in an isolated version of the software managed during code coverage evaluation.

#### 3.6.2.6 R Package `checked`

To assess whether an upgraded package introduces errors in existing packages, the developer tool, `checked` R package will be used to assess integration with an existing package ecosystem. Reverse dependency checking is comprised of re-evaluating `R CMD check` using the available cohort of packages before and after a package upgrade to look for any possibly broken package behaviors.

It's important to note that upgrades of dependencies in the package ecosystem may occur after this validation report is generated, and one would need to refer to the reverse dependency checks conducted as those dependencies were upgraded to review any changes to behaviors if identified. Although individual package validation is considered robust for most use cases, comprehensive review of package evaluation would require review of the test execution of a package and all of its dependencies.

### 3.6.3 Testing Steps

To evaluate an R package, a number of steps are taken to characterize the test environment, development best practices and efficacy of the R package's test suite in showing that the advertised behaviors are accurate. These steps include:

1. Documenting adherence to R package development best practices in `R CMD check` Results. Any errors encountered in this process will prevent the package from building properly. Nevertheless, errors disqualify a package for validation. The less severe messages, warnings and notes are also disqualifying, unless accompanied by developer commentary in the file `cran-comments.md`.
2. Evaluating the provided test suite, reporting the total number of tests as well as an itemization of passed and failed unit tests. No test failures are permitted for successful validation.
3. Evaluation of the test suite coverage, reporting the percentage of code which is executed during the evaluation of the testing suite. Coverage must exceed 80%, or package developers may set a realistic coverage threshold appropriate for their package which must be met for successful validation according to the tables below in Coverage Remediation.
4. Generation of a traceability matrix, which links unit test execution to the included narrative descriptions of provided behaviors. All requirements must be covered by at least one test for successful validation.
5. If a package already exists in the target environment and is to be updated, then re-evaluation of any reverse dependencies is also conducted prior to updates.

#### 3.6.3.1 Test Remediation

If a package fails to meet these criteria, the package must be manually approved. This validation document, though not sufficient for validating the package in that case, may be used to document the gaps that were identified. Justifications for why a package may still be of sufficient quality for validation despite failures to meet these testing criteria must be documented in the Remediation section of this document.

#### 3.6.3.2 Coverage Remediation

Coverage thresholds may not be applicable or realistic in all cases. The following are some categories of considerations when evaluating the applicability of coverage thresholds.

| | Coverage Consideration | Rationale |
|---|---|---|
| 1 | Overly Complex Testing | In some cases, additional code coverage requires overbearing test engineering which results in diminishing test quality. As examples, this can happen when a software is dependent on other systems or uses metaprogramming techniques that causes lines of code to be uncounted despite being evaluated. |
| 2 | Decision Impact | When packages are unlikely to impact critical decision making, then those packages may not require such stringent testing policy. |
| 3 | Adoption, Longevity | Packages may have few or no tests, but are nonetheless widely used. For example, old, externally developed packages may have been developed in a time when testing was less widespread. Nonetheless, such packages can have extremely broad adoption, either directly or as dependencies to other packages. In this scenario, the breadth of adoption and longevity of use can be used to justify an otherwise unacceptable code coverage. |
| 4 | Developer Trust | Packages may not meet coverage criteria, but are developed by established members of the R community or trusted institutions. |

Provided these considerations, below are guidelines for the relative applicability at various levels of test coverage which may be used as guidelines for remediation.

| Coverage (%) | Needs Justification | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 80+ | No | *NA* | *NA* | *NA* | *NA* |
| 40-80 | Yes | OK | OK | OK | OK |
| 20-40 | Yes | | OK | OK | OK |
| 0-20 | Yes | | | OK | OK |

#### 3.6.3.3 `R CMD check` Remediation

For external packages which prompt `R CMD check` `WARNING`s or `NOTE`s, and do not bundle a `cran-comments.md` file, an individual would need to review the output and provide a justification before such a package would be acceptable.

# 4 Validation Report

## 4.1 Software Specifications

### 4.1.1 Dependencies

R packages are dependent first and foremost on a functioning installation of the R language and a minimal set of R packages comprised of the R language standard library. Unless restricted, an R package is considered to work on all versions of the R language. The R language itself relies on a number of system libraries, which are considered out-of-scope for an individual R package's dependencies, and should be validated as part of the computing environment into which the provided R package is to be installed.

The `aNCA` R package relies on additional R packages or imposes language version restrictions as specified below:

| Type | Name | Version Restriction |
|---|---|---|
| Dependency | R | (>= 3.5) |
| Import | dplyr | *NA* |
| Import | formatters | *NA* |
| Import | ggplot2 | *NA* |
| Import | magrittr | *NA* |
| Import | methods | *NA* |
| Import | PKNCA | (>= 0.12.0) |
| Import | plotly | *NA* |
| Import | purrr | *NA* |
| Import | rlang | *NA* |
| Import | rlistings | *NA* |
| Import | stats | *NA* |
| Import | stringr | *NA* |
| Import | tern | *NA* |
| Import | tidyr | *NA* |
| Import | units | *NA* |
| Import | utils | *NA* |

Additional software dependencies are not required for general use, but may be required to build the R package, its documentation, or evaluatate its test suite. Such R packages are labelled "Suggests". Likewise, the package at hand might enhance the behavior of existing R packages with additional functionality. Packages enhanced by the R package at hand are labelled "Enhances". For further information about the distinctions of these classifications of related software, refer to section *Package Dependencies* of "Writing R Extensions".

| Type | Name | Version Restriction |
|---|---|---|
| Suggests | arrow | *NA* |
| Suggests | DT | *NA* |
| Suggests | ggh4x | *NA* |
| Suggests | haven | *NA* |
| Suggests | htmlwidgets | *NA* |
| Suggests | jsonlite | *NA* |
| Suggests | knitr | *NA* |
| Suggests | lintr | (>= 3.1.2) |
| Suggests | logger | *NA* |
| Suggests | markdown | *NA* |
| Suggests | nestcolor | *NA* |
| Suggests | openxlsx2 | *NA* |
| Suggests | pak | *NA* |
| Suggests | reactable | *NA* |

| Type | Name | Version Restriction |
|---|---|---|
| Suggests | reactable.extras | *NA* |
| Suggests | rmarkdown | *NA* |
| Suggests | sass | *NA* |
| Suggests | scales | *NA* |
| Suggests | shiny | *NA* |
| Suggests | shinycssloaders | *NA* |
| Suggests | shinyjs | *NA* |
| Suggests | shinyjqui | *NA* |
| Suggests | shinyWidgets | *NA* |
| Suggests | stringi | *NA* |
| Suggests | testthat | (>= 3.0.0) |
| Suggests | tools | *NA* |
| Suggests | vdiffr | *NA* |
| Suggests | withr | *NA* |
| Suggests | yaml | *NA* |

### 4.1.2 Release Notes

No documented developer updates provided

## 4.2 Software Version Control Summary

The `aNCA` R package source code is managed using the `git` version control system. The version of the software evaluated herein can be accessed via the following centrally managed code repository.

| `git` Attribute | Value |
|---|---|
| upstream url | `https://github.com/pharmaverse/aNCA.git` |
| version hash | 84a71ee58ece03e3aaf5337b0c48d47a4f24a65f |

## 4.3 Vulnerabilities scan results

The `OSV` database has been scanned looking for software vulnerabilities associated with the `aNCA` R package and its UNIX (Ubuntu) system dependencies. Identified vulnerabilities reports relevant to this validation are listed below:

*no discovered vulnerabilities*

## 4.4 Functional Requirements Specification

The software R package distributes a collection of user-facing functionality. This section itemizes each documented object within the R package and the associated documentation for each object within the R package.

Documentation provided with the R package can be made up of any number of subsections describing the behavior of a provided R objects. At a minimum, a documentation title is required, giving a brief overview of

behavior. For exceedingly simple or self-evident behaviors this may be sufficient. The detail of documentation is left to the discretion of the R package developers and should reflect the complexity of the functionality that they are offering and the amount of risk that that behavior might incur. This section includes documented behavior title, description and usage, as these sections are generally the most relevant for a general audience. Although not included within this report, the documentation provided with this functionality may include additional sections which explicitly itemize function parameters, provide further details, implementation or consideration notes, examples, and references.

For further details regarding the documentation format, please refer to section *Rd format* of "Writing R Extensions".

### 4.4.1 Apply Filters to a Dataset

```
apply_filters(data, filters)
```

#### 4.4.1.1 Description

This function applies a set of filters to a dataset. Each filter specifies a column, condition, and value to filter the dataset.

#### 4.4.1.2 Details

The function iterates over the list of filters and applies each filter to the dataset. The supported conditions are ==, >, <, >=, <= and !=.

### 4.4.2 Apply Labels to a dataset

```
apply_labels(data, labels_df, type)
```

#### 4.4.2.1 Description

This function adds "label" attributes to all columns in a dataset

### 4.4.3 Convert to Factor While Preserving Label

```
as_factor_preserve_label(x)
```

#### 4.4.3.1 Description

This function converts a vector to a factor while preserving its "label" attribute.

### 4.4.4 Calculate bioavailability with pivoted output

```
calculate_f(res_nca, f_aucs)
```

#### 4.4.4.1 Description

This function calculates bioavailability (F) based on AUC (Area Under Curve) data extracted from `res_nca`. It computes individual bioavailability where IV and EX data are available for a subject. If IV data is missing, it estimates bioavailability

using the mean IV values for that grouping. The output is pivoted such that each row represents all main results summarized for each profile in each subject. Columns are assumed to be in `%` units even if not explicitly stated.

### 4.4.5 Calculate Summary Statistics

```
calculate_summary_stats(data, input_groups = "NCA_PROFILE")
```

#### 4.4.5.1 Description

This function calculates various summary statistics for formatted output of PKNCA::pk.nca().

#### 4.4.5.2 Details

The function calculates the following statistics for numeric variables:

- Geometric mean (`geomean`)
- Geometric coefficient of variation (`geocv`)
- Arithmetic mean (`mean`)
- Standard deviation (`sd`)
- Minimum value (`min`)
- Maximum value (`max`)
- Median value (`median`)
- Count of missing values (`count.missing`)
- Count (`count`)

The resulting summary statistics are rounded to three decimal places. If units are different, they are standardized to the group's most frequent first unit.

### 4.4.6 Check overlap between existing and new slope rulesets

```
check_slope_rule_overlap(existing, new, slope_groups, .keep = FALSE)
```

#### 4.4.6.1 Description

Takes in tables with existing and incoming selections and exclusions, finds any overlap and differences, edits the ruleset table accordingly.

### 4.4.7 Create C0 Impute Column

```
create_start_impute(pknca_data)
```

#### 4.4.7.1 Description

Defines an impute column in the intervals of the PKNCAdata object based on data

### 4.4.8 Create duplicates in concentration data with Pre-dose and Last Values for Dosing Cycles

```
dose_profile_duplicates(
  conc_data,
  groups = c("USUBJID", "DOSNOA", "PARAM"),
  dosno = "DOSNOA",
  arrlt = "ARRLT",
  afrlt = "AFRLT",
  nrrlt = "NRRLT",
  nfrlt = "NFRLT"
)
```

#### 4.4.8.1 Description

This function duplicates and adjusts concentration data to ensure all dosing cycles have complete pre-dose and last concentration values. It is designed for use in pharmacokinetic analyses where dosing intervals and concentration values need to be aligned for each dose.

### 4.4.9 Exclude NCA Results Based on Parameter Thresholds

```
exclude_nca_by_param(
  parameter,
  min_thr = NULL,
  max_thr = NULL,
  affected_parameters = parameter
)
```

#### 4.4.9.1 Description

Exclude rows from NCA results based on specified thresholds for a given parameter. This function allows users to define minimum and/or maximum acceptable values for a parameter and excludes rows that fall outside these thresholds.

### 4.4.10 Export CDISC Data

```
export_cdisc(res_nca)
```

#### 4.4.10.1 Description

This function processes the results from a PKNCA and exports them into CDISC compliant datasets. Attention: All parameters that do no match pptest dataframe will be lost in this pipeline!

#### 4.4.10.2 Details

Outputs are the following:

- pknca_result Output from function call `pk.nca()` (formatted)
- pknca_result_raw Output from function call `pk.nca()` (needs to be merged with upper later on but now we avoid merge conflict)

### 4.4.11 Filter Breaks for X-Axis

```
filter_breaks(breaks = NA, plot = plot, min_cm_distance = 0.5, axis = "x")
```

#### 4.4.11.1 Description

Filters X-axis for consecutive breaks with at least the specified distance.

### 4.4.12 Filter dataset based on slope selections and exclusions

```
filter_slopes(data, slopes, profiles, slope_groups, check_reasons = FALSE)
```

#### 4.4.12.1 Description

This function filters main dataset based on provided slope selections an exclusions.

### 4.4.13 Flexible Violin/Box Plot

```
flexible_violinboxplot(
  boxplotdata,
  parameter,
  xvars,
  colorvars,
  varvalstofilter,
  columns_to_hover,
  box = TRUE,
  plotly = TRUE
)
```

#### 4.4.13.1 Description

This function generates a violin or box plot based on the provided data, parameter, and dose information.

### 4.4.14 Create PK Concentration Dataset

```
format_pkncaconc_data(
  ADNCA,
  group_columns,
  time_column = "AFRLT",
  rrlt_column = "ARRLT",
  route_column = "ROUTE"
)
```

#### 4.4.14.1 Description

This function creates a pharmacokinetic concentration dataset from the provided ADNCA data.

#### 4.4.14.2 Details

The function performs the following steps:

- Checks for required columns and data.
- Filters out rows with EVID = 0 and PARAMCD containing ”DOSE” (dosing data- not CDISC standard)
- Calculates `TIME_DOSE`as the time of dose reference by the PK sample
- Creates `DOSNOA` variable, sequential numbers based on time of dose
- Adds a 'std_route' column taking values ”intravascular” or ”extravascular”.
- Arranges the data by group_columns.

### 4.4.15 Create Dose Intervals Dataset

```
format_pkncadata_intervals(
  pknca_conc,
  pknca_dose,
  params = c("aucinf.obs", "aucint.last", "auclast", "cmax", "half.life", "tmax",
    "lambda.z", "lambda.z.n.points", "r.squared", "adj.r.squared", "lambda.z.time.first",
    "aucpext.obs", "aucpext.pred", "clast.obs", "cl.obs"),
  start_from_last_dose = TRUE
)
```

#### 4.4.15.1 Description

This function creates a dataset with dose intervals and specified pharmacokinetic parameters.

#### 4.4.15.2 Details

The function performs the following steps:

- Creates a vector with all pharmacokinetic parameters.
- Based on dose times, creates a data frame with start and end times.
- If TAU column is present in data, sets last dose end time to start + TAU
- If no TAU column in data, sets last dose end time to the time of last sample
- Adds logical columns for each specified parameter.

### 4.4.16 Create PK Dose Dataset

```
format_pkncadose_data(pkncaconc_data, group_columns)
```

#### 4.4.16.1 Description

This function creates a pharmacokinetic dose dataset from the provided concentration data.

#### 4.4.16.2 Details

The function performs the following steps:

- Arranges and groups the data by group_columns
- Selects the first row within each group (arranged by DOSNOA- a variable created in `format_pkncaconc_data`)

Note*: This function is designed to work with the output of `format_pkncaconc_data`.

### 4.4.17 Wrapper around aNCA::pkcg01() function. Calls the function with `LIN` scale argument.

```
g_pkcg01_lin(data, ...)
```

#### 4.4.17.1 Description

Wrapper around aNCA::pkcg01() function. Calls the function with `LIN` scale argument.

### 4.4.18 Wrapper around aNCA::pkcg01() function. Calls the function with `LOG` scale argument.

```
g_pkcg01_log(data, ...)
```

#### 4.4.18.1 Description

Wrapper around aNCA::pkcg01() function. Calls the function with `LOG` scale argument.

### 4.4.19 Wrapper around aNCA::pkcg02() function. Calls the function with `LIN` scale argument.

```
g_pkcg02_lin(data, ...)
```

#### 4.4.19.1 Description

Wrapper around aNCA::pkcg02() function. Calls the function with `LIN` scale argument.

### 4.4.20 Wrapper around aNCA::pkcg02() function. Calls the function with `LOG` scale argument.

```
g_pkcg02_log(data, ...)
```

#### 4.4.20.1 Description

Wrapper around aNCA::pkcg02() function. Calls the function with `LOG` scale argument.

### 4.4.21 Generate a General Line Plot for ADNCA Dataset

```
general_lineplot(
  data,
  selected_analytes,
  selected_pcspec,
```

```
  selected_usubjids,
  colorby_var,
  time_scale,
  yaxis_scale,
  show_threshold = FALSE,
  threshold_value = 0,
  cycle = NULL
)
```

#### 4.4.21.1 Description

This function generates a line plot for an ADNCA dataset based on user-selected analytes, subjects, and other parameters. The plot can be customized to display data on a linear or logarithmic scale and can be filtered by cycle.

#### 4.4.21.2 Details

The function performs the following steps:a

- Filters the data based on the selected analytes, matrices, and subjects.
- Selects relevant columns and removes rows with missing concentration values.
- Converts 'USUBJID', 'NCA_PROFILE', and 'DOSEA' to factors.
- Filters the data by cycle if `time_scale` is "By Cycle" while creating duplicates for predose samples if needed.
- Adjusts concentration values for logarithmic scale if `yaxis_scale` is "Log".
- Generates a line plot using the `g_ipp` function with the specified parameters.
- Adjusts the y-axis to logarithmic scale if `yaxis_scale` is "Log".

### 4.4.22 Generate a Mean Concentration Plot for ADNCA Dataset

```
general_meanplot(
  data,
  selected_studyids,
  selected_analytes,
  selected_pcspecs,
  selected_cycles,
  id_variable = "DOSEA",
  plot_ylog = FALSE,
  plot_sd = FALSE,
  plot_ci = FALSE
)
```

#### 4.4.22.1 Description

This function generates a mean concentration plot for an ADNCA dataset based on user-selected study IDs, analytes, and cycles. The plot can be customized to display data on a linear or logarithmic scale and can optionally include standard deviation error bars.

### 4.4.23 Transform Units

```
get_conversion_factor(initial_unit, target_unit)
```

#### 4.4.23.1 Description

This function transforms a value from an initial unit to a target unit.

### 4.4.24 Get the Label of a Heading

```
get_label(labels_df, variable, type)
```

#### 4.4.24.1 Description

This function retrieves the label of a heading from a labels file.

### 4.4.25 Check if a Vector Has a Label

```
has_label(x)
```

#### 4.4.25.1 Description

This function checks if a vector has a ”label” attribute.

### 4.4.26 Add specified imputation methods to the intervals in a PKNCAdata or data.frame object.

```
interval_add_impute(
  data,
  target_impute,
  after,
  target_params,
  target_groups,
  ...
)
```

#### 4.4.26.1 Description

Add specified imputation methods to the intervals in a PKNCAdata or data.frame object.

#### 4.4.26.2 Details

If already present the target_impute method will be added substituting the existing one. All new intervals created will be added right after their original ones.

### 4.4.27 Remove specified imputation from the intervals in a PKNCAdata or data.frame (intervals) object.

```
interval_remove_impute(data, target_impute, ...)
```

#### 4.4.27.1 Description

Remove specified imputation from the intervals in a PKNCAdata or data.frame (intervals) object.

### 4.4.28 Create PK Concentration Listing

```
l_pkconc(
  data,
  listgroup_vars = c("PARAM", "PCSPEC", "ROUTE"),
  grouping_vars = c("TRT01A", "USUBJID", "AVISIT"),
  displaying_vars = c("NFRLT", "AFRLT", "AVAL"),
  formatting_vars_table = NULL,
  title = paste0("Listing of PK Concentration by Treatment Group,",
    "Subject and Nominal Time, PK Population"),
  subtitle = NULL,
  footnote = "*: Subjects excluded from the summary table and mean plots"
)
```

#### 4.4.28.1 Description

This function creates a listing of pharmacokinetic (PK) concentration data segregating a dataset in lists that are customizable in title, footnotes, grouping/displayed variables, missing/zero values and/or number of digits displayed.

#### 4.4.28.2 Details

The function performs the following steps:

- Groups the data based on the specified grouping variables.
- Formats the 0 and NA values as defined by the formatting table.
- Creates a listing for each unique combination of the grouping variables.

The `formatting_vars_table` should be a data frame with the following columns:

- `var_name`: The name of the variable.
- `Label`: The label for the variable.
- `na_str`: The string to use for NA values.
- `zero_str`: The string to use for 0 values.
- `align`: The alignment for the variable (e.g., ”center”).
- `format_fun`: The formatting function to use (”round” or ”signif”).
- `digits`: The number of digits to use for numeric formatting.

### 4.4.29 Generate a Lambda Slope Plot

```
lambda_slope_plot(
  conc_pknca_df,
  row_values,
  myres = myres,
  r2adj_threshold = 0.7
)
```

#### 4.4.29.1 Description

This function generates a lambda slope plot using pharmacokinetic data. It calculates relevant lambda parameters and visualizes the data points used for lambda calculation, along with a linear regression line and additional plot annotations.

#### 4.4.29.2 Details

The function performs the following steps:

- Creates duplicates of the pre-dose and last doses of concentration data.
- Filters and arranges the input data to obtain relevant lambda calculation information.
- Identifies the data points used for lambda calculation.
- Calculates the fitness, intercept, and time span of the half-life estimate.
- Determines the subtitle color based on the R-squared adjusted value and half-life estimate.
- Generates a ggplot object with the relevant data points, linear regression line, and annotations.
- Converts the ggplot object to a plotly object for interactive visualization.

### 4.4.30 Calculate Matrix Ratios This function calculates the ratios for a given data set, based on the shared time points for each matrix concentration sample. The user can input multiple tissues for which ratios should be calculated.

```
multiple_matrix_ratios(
  data,
  matrix_col,
  conc_col,
  units_col,
  groups = c("NFRLT", "USUBJID"),
  spec1,
  spec2
)
```

#### 4.4.30.1 Description

The ratios are calculated as specimen 1 / specimen 2.

### 4.4.31 Parses annotations in the context of data. Special characters and syntax are substituted by actual data and/or substituted for format that is better parsed via rendering functions (e.g. plotly).

```
parse_annotation(data, text)
```

#### 4.4.31.1 Description

Parses annotations in the context of data. Special characters and syntax are substituted by actual data and/or substituted for format that is better parsed via rendering functions (e.g. plotly).

#### 4.4.31.2 Details

- `\n` character is substituted for `\<br\>` tag in order to add new lines in rendered image.

- `$COLNAME` is parsed to provide unique data value from the mentioned column.
- `!COLNAME` is parsed to provide `label` attribute for a given column name. If any values are missing from the provided data, they are substituted for `ERR` string.

### 4.4.32 Reshape PKNCA Results

```
pivot_wider_pknca_results(myres)
```

#### 4.4.32.1 Description

This function reshapes the structure of the results produced by the main function of the PKNCA package (pk.nca) in a way that each row represents all the main results summarized for each profile in each individual/subject. Excluding the ID variables, each column name corresponds with a calculated parameter and between brackets its corresponding units. AUC intervals, if present, are be added as additional columns.

### 4.4.33 Generate PK Concentration-Time Profile Plots

```
pkcg01(
  adpc = data(),
  xvar = "AFRLT",
  yvar = "AVAL",
  xvar_unit = "RRLTU",
  yvar_unit = "AVALU",
  color_var = NULL,
  color_var_label = NULL,
  xbreaks_var = "NFRLT",
  xbreaks_mindist = 0.5,
  xmin = NA,
  xmax = NA,
  ymin = NA,
  ymax = NA,
  xlab = paste0("!", xvar, " [$", xvar_unit, "]"),
  ylab = paste0("!", yvar, " [$", yvar_unit, "]"),
  title = NULL,
  subtitle = NULL,
  footnote = NULL,
  plotgroup_vars = c("ROUTE", "PCSPEC", "PARAM", "USUBJID"),
  plotgroup_names = list(ROUTE = "Route", PCSPEC = "Specimen", PARAM = "Analyte", USUBJID
    = "Subject ID"),
  scale = c("LIN", "LOG", "SBS")[1],
  studyid = "STUDYID",
  trt_var = "TRT01A",
  plotly = TRUE
)
```

#### 4.4.33.1 Description

This function generates a list of ggplots for PK concentration-time profiles.

### 4.4.34 Generate Combined PK Concentration-Time Profile Plot by Cohort

```
pkcg02(
  adpc = data(),
  xvar = "AFRLT",
  yvar = "AVAL",
  xvar_unit = "RRLTU",
  yvar_unit = "AVALU",
  color_var = NULL,
  color_var_label = NULL,
  xbreaks_var = "NFRLT",
  xbreaks_mindist = 0.5,
  xmin = NA,
  xmax = NA,
  ymin = NA,
  ymax = NA,
  xlab = paste0("!", xvar, " [$", xvar_unit, "]"),
  ylab = paste0("!", yvar, " [$", yvar_unit, "]"),
  title = NULL,
  subtitle = NULL,
  footnote = NULL,
  plotgroup_vars = c("ROUTE", "PCSPEC", "PARAM", "TRT01A"),
  plotgroup_names = list(ROUTE = "Route", PCSPEC = "Specimen", PARAM = "Analyte", TRT01A
    = "Treatment"),
  scale = c("LIN", "LOG", "SBS")[1],
  studyid = "STUDYID",
  trt_var = "TRT01A",
  plotly = TRUE
)
```

#### 4.4.34.1 Description

This function generates a list of plotly objects PK concentration-time profiles by group

### 4.4.35 Build Units Table for PKNCA

```
PKNCA_build_units_table(o_conc, o_dose)
```

#### 4.4.35.1 Description

This function generates a PKNCA units table including the potential unit segregating columns among the dose and/or concentration groups.

#### 4.4.35.2 Details

The function performs the following steps:

1. Ensures the unit columns (e.g., `concu`, `timeu`, `doseu`, `amountu`) exist in the inputs.
2. Joins the concentration and dose data based on their grouping columns.
3. Generates a PKNCA units table for each group, including conversion factors and custom units.
4. Returns a unique table with relevant columns for PKNCA analysis.

### 4.4.36 Calculate bioavailability for intravascular vs extravascular aucs

```
pknca_calculate_f(res_nca, f_aucs)
```

#### 4.4.36.1 Description

This function calculates bioavailability (F) based on AUC (Area Under Curve) data extracted from `res_nca`. It computes individual bioavailability where IV and EX data are available for a subject. If IV data is missing, it estimates bioavailability using the mean IV values for that grouping.

#### 4.4.36.2 Details

- The function extracts AUC data from `res_nca$data$conc$data` and filters for selected AUC types.
- It separates data into intravascular (IV) and extravascular (EX) groups.
- Individual bioavailability is calculated for subjects with both IV and EX data using PKNCA function `pk.calc.f`.
- If IV data is missing for a subject, the function estimates bioavailability using mean IV values for that grouping.
- The final output includes bioavailability estimates for individual subjects and mean-based estimates.

### 4.4.37 Calculates results for PKNCA analysis.

```
PKNCA_calculate_nca(pknca_data)
```

#### 4.4.37.1 Description

Calculates results for PKNCA analysis.

#### 4.4.37.2 Details

This function+ calculates results for PKNCA analysis using `PKNCA::pk.nca()`. It then joins the results with the dosing data, to create a full results data frame with the start and end times for each dose, from first and most recent dose.

### 4.4.38 Creates a `PKNCA::PKNCAdata` object.

```
PKNCA_create_data_object(adnca_data)
```

#### 4.4.38.1 Description

Creates a `PKNCA::PKNCAdata` object.

#### 4.4.38.2 Details

This function creates a standard PKNCAdata object from ADNCA data. It requires the following columns in the ADNCA data:

- STUDYID: Study identifier.
- PCSPEC: Matrix.
- ROUTE: Route of administration.

- DRUG: Drug identifier.
- USUBJID: Unique subject identifier.
- NCA_PROFILE: (Non- standard column). Can be any column, used for filtering the data for NCA
- PARAM: Analyte.
- AVAL: Analysis value.
- AVALU: AVAL unit.
- DOSEA: Dose amount.
- DOSEU: Dose unit.
- AFRLT: Actual time from first dose.
- ARRLT: Actual time from reference dose.
- NFRLT: Nominal time from first dose.
- ADOSEDUR: Duration of dose.
- RRLTU: Time unit.

1. Creating pk concentration data using `format_pkncaconc_data()`.
2. Creating dosing data using `format_pkncadose_data()`.
3. Creating `PKNCAconc` object using `PKNCA::PKNCAconc()`. with formula `AVAL ~ TIME | STUDYID + PCSPEC + DRUG + USUBJID / PARAM`.
4. Creating PKNCAdose object using `PKNCA::PKNCAdose()`. with formula `DOSEA ~ TIME | STUDYID + DRUG + USUBJID`.
5. Creating PKNCAdata object using `PKNCA::PKNCAdata()`.
6. Updating units in PKNCAdata object so each analyte has its own unit.

### 4.4.39 This function imputes the start concentration using the first concentration after dose

```
PKNCA_impute_method_start_c1(conc, time, start, end, ..., options = list())
```

#### 4.4.39.1 Description

This function imputes the start concentration using the first concentration after dose

#### 4.4.39.2 Details

This function adheres to the structure required by the `PKNCA` package to work with its functionalities.For more information, see the PKNCA Data Imputation Vignette.

### 4.4.40 This function imputes the start concentration using the log slope method.

```
PKNCA_impute_method_start_logslope(
  conc,
  time,
  start,
  end,
  ...,
  options = list()
)
```

#### 4.4.40.1 Description

This function imputes the start concentration using the log slope method.

#### 4.4.40.2 Details

This function adheres to the structure required by the `PKNCA` package to work with its functionalities. For more information, see the PKNCA Data Imputation Vignette.

### 4.4.41 Create a PKNCAdata Object for NCA or Slope Analysis

```
PKNCA_update_data_object(
  adnca_data,
  auc_data,
  method,
  selected_analytes,
  selected_profile,
  selected_pcspec,
  params,
  should_impute_c0 = TRUE
)
```

#### 4.4.41.1 Description

This function updates a previously prepared `PKNCAdata` object based on user selections for method, analyte, dose, specimen, and parameters.

#### 4.4.41.2 Details

Step 1: Update units in the `PKNCAdata` object ensuring unique analytes have their unique units

Step 2: Set `PKNCAoptions` for NCA calculation

Step 3: Format intervals using `format_pkncadata_intervals()`

Step 4: Apply filtering based on user selections and partial aucs

Step 5: Impute start values if requested

Note*: The function assumes that the `adnca_data` object has been created using the `PKNCA_create_data_object()` function.

### 4.4.42 Reads PK datasets from various file formats.

```
read_pk(path)
```

#### 4.4.42.1 Description

Reads PK datasets from various file formats.

#### 4.4.42.2 Details

Currently supported file formats include:

- `csv`
- `rds`

- xlsx
- sas7bdat
- xpt
- parquet

#### 4.4.43 Run the Shiny app

```
run_app(...)
```

##### 4.4.43.1 Description

Run the Shiny app

#### 4.4.44 Set an Empty Label if None Exists

```
set_empty_label(x)
```

##### 4.4.44.1 Description

This function sets an empty "label" attribute for a vector if it does not already have one.

#### 4.4.45 Translate Terms from One Nomenclature to Another

```
translate_terms(
  input_terms,
  mapping_col = "PKNCA",
  target_col = "PPTESTCD",
  metadata = pknca_cdisc_terms
)
```

##### 4.4.45.1 Description

This function translates a character vector of terms from one nomenclature to another using a mapping file.

## 4.5 Testing Infrastructure

Testing consists of two steps, which combine to test the required validation criteria. Two separate tools are used in these steps, `autovalidateutils` and `autovalidate`. `autovalidateutils` provides a toolkit for executing common testing procedures for the R package, and formats the output. `autovalidate` uses these files to perform additional checks and to populate a report based on a report template.

Specifically, `autovalidateutils` will output the following pieces of metadata. This tool is not required to produce these files, as all of the information collected could be compiled using standard tools. This tool is used for convenience and consistency of these files. The files that are produced include:

| Filename | Required | Content Description |
|---|---|---|
| coverage/sonarqube.xml | Yes | A sonarqube-formatted coverage results file |

| Filename | Required | Content Description |
|---|---|---|
| description.yml | Yes | A yaml-formatted version of the standard R package `DESCRIPTION` file |
| errors.yml | No | A yaml-formatted list of error messages to relay to `autovalidate` |
| news.md | No | A markdown-formatted version of the standard R package `NEWS` file |
| r_cmd_check.yml | Yes | A yaml-formatted output of R's `R CMD check` |
| session.yml | Yes | A yaml-formatted output of various diagnostic R functions used to characterize the testing environment and execution session |
| coverage/testing_session.yml | Yes | A yaml-formatted output of R functions used to characterize the R testing session |
| tests.Rout | Yes | Plain text output produced by R while executing the target package's test suite |
| traceability_matrix/mapper.yml | Yes | A yaml-formatted mapping of requirements and tests which evaluate them |
| traceability_matrix/junit.xml | Yes | A JUnit-formatted test results file |
| traceability_matrix/documentation/*.md | Yes | Any number of markdown-formatted documented requirements |
| version_control.yml | Yes | A yaml-formatted description of unique source code identifiers |

For further details about the expected structure or contents, please refer to the source template files and generating functions in `autovalidateutils`.

## 4.6 Test Cases & Report

### 4.6.1 Installation Verification

Verification of the ability to install the R package is fulfilled by successful execution of the below *Section "`R CMD check` Results"*. Within this section, the console output of the install procedure is included and the loading of the R package for evaluation is conducted. For further details and a log of this process, please refer to the above section.

Installation verification is managed by the R language, which checks for the ability to successfully load the R package from a temporary, testing and final library location. Upon successful verification of installation, the following execution log will end with `DONE`.

#### 4.6.1.1 Installation System Information

```
> sessionInfo()
R version 4.4.3 (2025-02-28)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 22.04.5 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.10.0
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
```

```
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: Etc/UTC
tzcode source: system (glibc)

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
 [1] jsonlite_2.0.0          compiler_4.4.3          brio_1.1.5
 [4] xml2_1.3.8              covr_3.6.4.9005         git2r_0.36.2
 [7] rcmdcheck_1.4.0         callr_3.7.6             autovalidateutils_2.3.1
[10] yaml_2.3.10             R6_2.6.1                covtracer_0.0.1
[13] pak_0.8.0.1             curl_6.2.2              igraph_2.1.4
[16] knitr_1.50              desc_1.4.3              rprojroot_2.0.4
[19] rlang_1.1.5             testthat_3.2.3          stringi_1.8.7
[22] xfun_0.52               lazyeval_0.2.2          rex_1.2.1
[25] cli_3.6.4               withr_3.0.2             magrittr_2.0.3
[28] options_0.3.1           ps_1.9.0                digest_0.6.37
[31] processx_3.8.6          remotes_2.4.2.9000      lifecycle_1.0.4
[34] prettyunits_1.2.0       evaluate_1.0.3          rd2markdown_0.1.2
[37] xopen_1.0.1             checked_0.2.8.9000      pkgbuild_1.4.7
[40] httr_1.4.7              tools_4.4.3             pkgconfig_2.0.3

> capabilities(Xchk = FALSE)
       jpeg         png        tiff       tcltk         X11        aqua
       TRUE        TRUE        TRUE        TRUE          NA       FALSE
   http/ftp     sockets      libxml        fifo      cledit       iconv
       TRUE        TRUE       FALSE        TRUE       FALSE        TRUE
        NLS       Rprof     profmem       cairo         ICU long.double
       TRUE        TRUE        TRUE        TRUE        TRUE        TRUE
    libcurl
       TRUE

> .libPaths()
[1] "/tmp/Rtmp37vTTP/R_LIB_validation"   "/tmp/Rtmp37vTTP/R_LIB_dependencies"
[3] "/usr/local/lib/R/site-library"      "/usr/lib/R/site-library"
[5] "/usr/lib/R/library"                 "/opt/autovalidate-r/lib"
```

Itemized installed packages and versions for all dependencies used directly or indirectly for testing:

```
  [1] abind (1.4-8)             arrow (17.0.0.1)
  [3] askpass (1.2.1)           assertthat (0.2.1)
  [5] autovalidateutils (2.3.1) backports (1.5.0)
  [7] base64enc (0.1-3)         bit (4.5.0)
  [9] bit64 (4.5.2)             boot (1.3-31)
 [11] brio (1.1.5)              broom (1.0.8)
 [13] bslib (0.8.0)             cachem (1.1.0)
 [15] callr (3.7.6)             car (3.1-3)
 [17] carData (3.0-5)           checked (0.2.8.9000)
 [19] checkmate (2.3.2)         cli (3.6.3)
 [21] clipr (0.8.0)             codetools (0.2-20)
 [23] colorspace (2.1-1)        commonmark (1.9.2)
 [25] compiler (4.4.3)          covr (3.6.4.9005)
```

```
 [27] covtracer (0.0.1)         cowplot (1.1.3)
 [29] cpp11 (0.5.0)             crayon (1.5.3)
 [31] crosstalk (1.2.1)         curl (5.2.3)
 [33] cyclocomp (1.1.1)         data.table (1.16.0)
 [35] Deriv (4.1.6)             desc (1.4.3)
 [37] diffobj (0.3.5)           digest (0.6.37)
 [39] doBy (4.6.24)             dplyr (1.1.4)
 [41] DT (0.33)                 emmeans (1.10.4)
 [43] estimability (1.5.1)      evaluate (1.0.1)
 [45] fansi (1.0.6)             farver (2.1.2)
 [47] fastmap (1.2.0)           fontawesome (0.5.2)
 [49] forcats (1.0.0)           formatters (0.5.11)
 [51] Formula (1.2-5)           fs (1.6.4)
 [53] generics (0.1.3)          ggh4x (0.3.0)
 [55] ggplot2 (3.5.1)           git2r (0.36.2)
 [57] glue (1.8.0)              graphics (4.4.3)
 [59] grDevices (4.4.3)         grid (4.4.3)
 [61] gridExtra (2.3)           gtable (0.3.5)
 [63] haven (2.5.4)             highr (0.11)
 [65] hms (1.1.3)               htmltools (0.5.8.1)
 [67] htmlwidgets (1.6.4)       httpuv (1.6.15)
 [69] httr (1.4.7)              igraph (2.1.4)
 [71] isoband (0.2.7)           jquerylib (0.1.4)
 [73] jsonlite (1.8.9)          knitr (1.48)
 [75] labeling (0.4.3)          later (1.3.2)
 [77] lattice (0.22-7)          lazyeval (0.2.2)
 [79] lifecycle (1.0.4)         lintr (3.1.2)
 [81] lme4 (1.1-37)             logger (0.4.0)
 [83] magrittr (2.0.3)          markdown (1.13)
 [85] MASS (7.3-65)             Matrix (1.7-3)
 [87] MatrixModels (0.5-3)      memoise (2.0.1)
 [89] methods (4.4.3)           mgcv (1.9-3)
 [91] microbenchmark (1.5.0)    mime (0.12)
 [93] minqa (1.2.8)             modelr (0.1.11)
 [95] munsell (0.5.1)           mvtnorm (1.3-1)
 [97] nestcolor (0.1.3)         nlme (3.1-168)
 [99] nloptr (2.1.1)            nnet (7.3-20)
[101] numDeriv (2016.8-1.1)     openssl (2.2.2)
[103] openxlsx2 (1.15)          options (0.3.1)
[105] pak (0.8.0.2)             parallel (4.4.3)
[107] pbkrtest (0.5.3)          pillar (1.9.0)
[109] pkgbuild (1.4.4)          pkgconfig (2.0.3)
[111] pkgload (1.4.0)           PKNCA (0.12.0)
[113] plotly (4.11.0)           praise (1.0.0)
[115] prettyunits (1.2.0)       processx (3.8.4)
[117] progress (1.2.3)          promises (1.3.0)
[119] ps (1.8.0)                purrr (1.0.2)
[121] quantreg (5.98)           R6 (2.5.1)
[123] rappdirs (0.3.3)          rbibutils (2.3)
[125] rcmdcheck (1.4.0)         RColorBrewer (1.1-3)
[127] Rcpp (1.0.13)             RcppEigen (0.3.4.0.2)
[129] rd2markdown (0.1.2)       Rdpack (2.6.1)
[131] reactable (0.4.4)         reactable.extras (0.2.1)
[133] reactR (0.6.1)            readr (2.1.5)
[135] reformulas (0.4.0)        rematch2 (2.1.2)
[137] remotes (2.4.2.9000)      rex (1.2.1)
```

```
[139] rjson (0.2.23)            rlang (1.1.4)
[141] rlistings (0.2.11)        rmarkdown (2.28)
[143] rprojroot (2.0.4)         rtables (0.6.12)
[145] sass (0.4.9)              scales (1.3.0)
[147] sessioninfo (1.2.3)       shiny (1.9.1)
[149] shinycssloaders (1.1.0)   shinyjqui (0.4.1)
[151] shinyjs (2.1.0)           shinyWidgets (0.9.0)
[153] sourcetools (0.1.7-1)     SparseM (1.84-2)
[155] splines (4.4.3)           stats (4.4.3)
[157] stringi (1.8.4)           stringr (1.5.1)
[159] survival (3.8-3)          sys (3.4.3)
[161] tern (0.9.8)              testthat (3.2.1.1)
[163] tibble (3.2.1)            tidyr (1.3.1)
[165] tidyselect (1.2.1)        tinytex (0.57)
[167] tools (4.4.3)             tzdb (0.4.0)
[169] units (0.8-7)             utf8 (1.2.4)
[171] utils (4.4.3)             vctrs (0.6.5)
[173] vdiffr (1.0.8)            viridisLite (0.4.2)
[175] vroom (1.6.5)             waldo (0.5.3)
[177] withr (3.0.1)             xfun (0.52)
[179] xml2 (1.3.6)              xmlparsedata (1.0.5)
[181] xopen (1.0.1)             xtable (1.8-4)
[183] yaml (2.3.10)             zip (2.3.1)
```

#### 4.6.1.2 Installation Execution Log

```
* installing *source* package ‘aNCA’ ...
** using staged installation
** R
** data
*** moving datasets to lazyload DB
** inst
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
** building package indices
** installing vignettes
** testing if installed package can be loaded from temporary location
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation path
* creating tarball
packaged installation of ‘aNCA’ as ‘aNCA_0.0.0.9095_R_x86_64-pc-linux-gnu.tar.gz’
* DONE (aNCA)
```

Upon successful installation, the R package can be expected to install correctly using the methods described in *Section "Package Installation"*.

### 4.6.2 `R CMD check` Results

The built-in R facility for evaluating a R package for valid structure and adherence to R package development best practices, `R CMD check`, was executed against the target R package.

The evaluation environment used had the following environment attributes.

| Environment Attribute | Value |
|---|---|
| R Version | 4.4.3 |
| Platform | using platform: x86_64-pc-linux-gnu |

The evaluation of `R CMD check` produced the following output.

```
* using log directory ‘/builds/autovalidate-r/package-
submission/portal/validation_staging/generated/check/aNCA.Rcheck’
* using R version 4.4.3 (2025-02-28)
* using platform: x86_64-pc-linux-gnu
* R was compiled by
    gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
    GNU Fortran (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
* running under: Ubuntu 22.04.5 LTS
* using session charset: UTF-8
* using options ‘--no-manual --ignore-vignettes’
* checking for file ‘aNCA/DESCRIPTION’ ... OK
* this is package ‘aNCA’ version ‘0.0.0.9095’
* package encoding: UTF-8
* checking package namespace information ... OK
* checking package dependencies ... OK
* checking if this is a source package ... OK
* checking if there is a namespace ... OK
* checking for executable files ... OK
* checking for hidden files and directories ... OK
* checking for portable file names ... OK
* checking for sufficient/correct file permissions ... OK
* checking whether package ‘aNCA’ can be installed ... OK
* checking installed package size ... OK
* checking package directory ... OK
* checking DESCRIPTION meta-information ... OK
* checking top-level files ... OK
* checking for left-over files ... OK
* checking index information ... OK
* checking package subdirectories ... OK
* checking code files for non-ASCII characters ... OK
* checking R files for syntax errors ... OK
* checking whether the package can be loaded ... OK
* checking whether the package can be loaded with stated dependencies ... OK
* checking whether the package can be unloaded cleanly ... OK
* checking whether the namespace can be loaded with stated dependencies ... OK
* checking whether the namespace can be unloaded cleanly ... OK
* checking loading without being on the library search path ... OK
* checking whether startup messages can be suppressed ... OK
* checking dependencies in R code ... OK
* checking S3 generic/method consistency ... OK
* checking replacement functions ... OK
* checking foreign function calls ... OK
* checking R code for possible problems ... OK
* checking Rd files ... OK
* checking Rd metadata ... OK
* checking for missing documentation entries ... OK
* checking for code/documentation mismatches ... OK
* checking Rd \usage sections ... OK
```

```
* checking Rd contents ... OK
* checking for unstated dependencies in examples ... OK
* checking contents of ‘data’ directory ... OK
* checking data for non-ASCII characters ... OK
* checking LazyData ... OK
* checking data for ASCII and uncompressed saves ... OK
* checking installed files from ‘inst/doc’ ... OK
* checking files in ‘vignettes’ ... SKIPPED
* checking examples ... OK
* checking for unstated dependencies in ‘tests’ ... OK
* checking tests ...
  Running ‘testthat.R’
 ERROR
Running the tests in ‘tests/testthat.R’ failed.
Last 13 lines of output:
  ── Failure ('test-flexible_violinboxplot.R:78:5'): flexible_violinboxplot:
creates a violin plot when box = FALSE ──
  Snapshot of `testcase` to 'flexible_violinboxplot/violin-plot.svg' has changed
  * Download and unzip run artifact
  * Copy 'tests/testthat/_snaps/flexible_violinboxplot/violin-plot.new.svg' to
local test directory
  * Run `testthat::snapshot_review('flexible_violinboxplot/')` to review changes
  Backtrace:
      ▆
   1. └─vdiffr::expect_doppelganger("violin_plot", violin_plot) at test-
flexible_violinboxplot.R:78:5
   2.   ├─base::withCallingHandlers(...)
   3.   └─testthat::expect_snapshot_file(...)

  [ FAIL 1 | WARN 0 | SKIP 0 | PASS 547 ]
  Error: Test failures
  In addition: There were 12 warnings (use warnings() to see them)
  Execution halted
Running the tests in ‘tests/vdiffr.[rR]’ failed.
Last 13 lines of output:
    'spacingAndGlyphs'>CMAX  [ ng/mL ]</text>
    <rect x='627.13' y='253.52' width='87.39' height='60.85' style='stroke-
width:
    1.07; stroke: none; fill: #FFFFFF;' />
  @@ 69,5 / 72,5 @@
    <text x='655.37' y='286.00' style='font-size: 8.80px; font-family: sans;'
text
    Length='4.89px' lengthAdjust='spacingAndGlyphs'>1</text>
    <text x='655.37' y='303.28' style='font-size: 8.80px; font-family: sans;'
text
    Length='4.89px' lengthAdjust='spacingAndGlyphs'>2</text>
  < <text x='32.79' y='14.56' style='font-size: 13.20px; font-family: sans;'
textL
  : ength='58.70px' lengthAdjust='spacingAndGlyphs'>violin_plot</text>
  > <text x='27.90' y='14.56' style='font-size: 13.20px; font-family: sans;'
textL
  : ength='58.70px' lengthAdjust='spacingAndGlyphs'>violin_plot</text>
    </g>
    </svg>

* DONE
```

```
Status: 1 ERROR
See
  ‘/builds/autovalidate-r/package-
submission/portal/validation_staging/generated/check/aNCA.Rcheck/00check.log’
for details.
```

#### 4.6.2.1 `R CMD check` Errors

The evaluation of `R CMD check` produced the following error messages.

```
checking tests ...
  Running ‘testthat.R’
 ERROR
Running the tests in ‘tests/testthat.R’ failed.
Last 13 lines of output:
  ── Failure ('test-flexible_violinboxplot.R:78:5'): flexible_violinboxplot:
creates a violin plot when box = FALSE ──
  Snapshot of `testcase` to 'flexible_violinboxplot/violin-plot.svg' has changed
  * Download and unzip run artifact
  * Copy 'tests/testthat/_snaps/flexible_violinboxplot/violin-plot.new.svg' to
local test directory
  * Run `testthat::snapshot_review('flexible_violinboxplot/')` to review changes
  Backtrace:
      ▆
   1. └─vdiffr::expect_doppelganger("violin_plot", violin_plot) at test-
flexible_violinboxplot.R:78:5
   2.   ├─base::withCallingHandlers(...)
   3.   └─testthat::expect_snapshot_file(...)

  [ FAIL 1 | WARN 0 | SKIP 0 | PASS 547 ]
  Error: Test failures
  In addition: There were 12 warnings (use warnings() to see them)
  Execution halted
Running the tests in ‘tests/vdiffr.[rR]’ failed.
Last 13 lines of output:
    'spacingAndGlyphs'>CMAX  [ ng/mL ]</text>
    <rect x='627.13' y='253.52' width='87.39' height='60.85' style='stroke-
width:
    1.07; stroke: none; fill: #FFFFFF;' />
  @@ 69,5 / 72,5 @@
    <text x='655.37' y='286.00' style='font-size: 8.80px; font-family: sans;'
text
    Length='4.89px' lengthAdjust='spacingAndGlyphs'>1</text>
    <text x='655.37' y='303.28' style='font-size: 8.80px; font-family: sans;'
text
    Length='4.89px' lengthAdjust='spacingAndGlyphs'>2</text>
  < <text x='32.79' y='14.56' style='font-size: 13.20px; font-family: sans;'
textL
  : ength='58.70px' lengthAdjust='spacingAndGlyphs'>violin_plot</text>
  > <text x='27.90' y='14.56' style='font-size: 13.20px; font-family: sans;'
textL
  : ength='58.70px' lengthAdjust='spacingAndGlyphs'>violin_plot</text>
    </g>
```

```
</svg>
```

## 4.7 Test Cases

### 4.7.1 Test Execution Information

Tests were executed on a system with the following specifications.

| Environment | Specification |
|---|---|
| Operating System | Ubuntu 22.04.5 LTS |
| Platform | x86_64-pc-linux-gnu |
| System | x86_64, linux-gnu |
| Execution Date & Time | 2025-06-24 12:52:12 GMT |

To ensure maximum control over the testing environment we execute tests in the separate R process where only packages required by the testing infrastructure have been loaded. The following information about the R session was captured during the runtime of the testing process.

```
R version 4.4.3 (2025-02-28)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 22.04.5 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.10.0
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: Etc/UTC
tzcode source: system (glibc)

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] dplyr_1.1.4      testthat_3.2.1.1 aNCA_0.0.0.9095

loaded via a namespace (and not attached):
 [1] tidyselect_1.2.1   viridisLite_0.4.2  farver_2.1.2       arrow_17.0.0.1
 [5] fastmap_1.2.0      lazyeval_0.2.2     vdiffr_1.0.8       rex_1.2.1
 [9] digest_0.6.37      lifecycle_1.0.4    waldo_0.5.3        survival_3.8-3
[13] magrittr_2.0.3     compiler_4.4.3     rlang_1.1.4        tools_4.4.3
[17] yaml_2.3.10        utf8_1.2.4         data.table_1.16.0  labeling_0.4.3
[21] htmlwidgets_1.6.4  bit_4.5.0          xml2_1.3.6         RColorBrewer_1.1-3
```

```
[25] covr_3.6.4.9005    withr_3.0.1        purrr_1.0.2        rlistings_0.2.11
[29] desc_1.4.3         grid_4.4.3         ggh4x_0.3.0        fansi_1.0.6
[33] openxlsx2_1.15     diffobj_0.3.5      colorspace_2.1-1   PKNCA_0.12.0
[37] nestcolor_0.1.3    ggplot2_3.5.1      scales_1.3.0       cli_3.6.3
[41] crayon_1.5.3       generics_0.1.3     httr_1.4.7         tzdb_0.4.0
[45] stringr_1.5.1      splines_4.4.3      assertthat_0.2.1   vctrs_0.6.5
[49] Matrix_1.7-3       jsonlite_1.8.9     tern_0.9.8         hms_1.1.3
[53] bit64_4.5.2        crosstalk_1.2.1    plotly_4.11.0      tidyr_1.3.1
[57] units_0.8-7        rtables_0.6.12     formatters_0.5.11  glue_1.8.0
[61] stringi_1.8.4      gtable_0.3.5       munsell_0.5.1      tibble_3.2.1
[65] pillar_1.9.0       htmltools_0.5.8.1  brio_1.1.5         reactable_0.4.4
[69] R6_2.5.1           Rdpack_2.6.1       lattice_0.22-7     haven_2.5.4
[73] readr_2.1.5        rbibutils_2.3      backports_1.5.0    broom_1.0.8
[77] Rcpp_1.0.13        zip_2.3.1          nlme_3.1-168       checkmate_2.3.2
[81] mgcv_1.9-3         forcats_1.0.0      pkgconfig_2.0.3
```

The testing process is running exactly the same R installation, meaning it inherits the system and R properties listed in the section 4.5.1.1. Differences that could be meaningful were included in the description above.

### 4.7.2 Test Execution Log

The following code and output was captured during the executing of the R package test suite.

```
R version 4.4.3 (2025-02-28) -- \"Trophy Case\"
Copyright (C) 2025 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type \'license()\' or \'licence()\' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type \'contributors()\' for more information and
\'citation()\' on how to cite R or R packages in publications.

Type \'demo()\' for some demos, \'help()\' for on-line help, or
\'help.start()\' for an HTML browser interface to help.
Type \'q()\' to quit R.

> library(\'aNCA\')
Registered S3 method overwritten by \'tern\':
  method   from
  tidy.glm broom
> options(width = 80L, testthat.progress.max_fails = 1L, testthat.use_colours =
FALSE,
+     cli.unicode = FALSE, crayon.enabled = FALSE, useFancyQuotes = FALSE)
> testthat::test_package(\"aNCA\", stop_on_failure = FALSE, reporter = {
+     testthat::MultiReporter$new(reporters =
list(testthat::SummaryReporter$new(file = \"/builds/autovalidate-r/package-submi
ssion/portal/validation_staging/generated/coverage/logs/tests_summary.log\",
+         show_praise = FALSE, omit_dots = TRUE, max_reports = Inf),
+         testthat::JunitReporter$new(file = \"/builds/autovalidate-r/package-
submission/portal/validation_staging/generated/traceability_matrix/junit.xml\"),
```

```
+         testthat::LocationReporter$new(file =
\"/builds/autovalidate-r/package-submission/portal/validation_staging/generated/
coverage/logs/error_locations.log\")))
+ })
Joining with `by = join_by(NCA_PROFILE)`
Loading required namespace: ggh4x

Attaching package: \'dplyr\'

The following object is masked from \'package:testthat\':

    matches

The following objects are masked from \'package:stats\':

    filter, lag

The following objects are masked from \'package:base\':

    intersect, setdiff, setequal, union

There were 12 warnings (use warnings() to see them)
> yaml::write_yaml(x = paste0(tryCatch(capture.output(sessionInfo()),
+     error = function(e) list(as.character(e))), collapse = \"\\n\"),
+     file = \"/builds/autovalidate-r/package-
submission/portal/validation_staging/generated/coverage/testing_session.yml\")
> write(jsonlite::toJSON(rapply(tryCatch(sessionInfo(), error = function(e)
list(as.character(e))),
+     unclass, how = \"list\"), auto_unbox = TRUE, pretty = TRUE),
+     file.path(\"/builds/autovalidate-r/package-
submission/portal/validation_log\",
+         \"testing_sessioninfo.json\"))
>
```

### 4.7.3 Testing Summary

| Total Tests Cases | Passed | Failed | Error | Unknown |
|---|---|---|---|---|
| 571 | 570 (100%) | 1 (0%) | 0 (0%) | 0 (0%) |

The *Failed* category aggregates tests that successfully completed but did not meet expectations, while the *Error* category consists of tests that raised an error before completion. The *Unknown* category is reserved for tests where the status could not be properly assessed due to limitations of the testing framework used by the package. Such a situation might occur when we identify tests failures in a package using a testing framework which cannot properly report which particular test did not meet expectations. Therefore, for those frameworks, if any tests failure is identified we indicate the inconclusive nature of these testing frameworks by marking all tests as *Unknown*. Further investigation of the impact of test failures will be remarked on in remediation comments.

Note that a test may be comprised of multiple test cases, which together evaluate a described behavior. Likewise, some tests may not evaluate user-facing requirement functionality. Therefore, the total number of tests and overall test coverage may exceed what is represented in the traceability matrix.

### 4.7.4 Tests coverage

To ensure a sufficient portion of the package's codebase is covered by unit-tests, the code execution is traced, documenting the code coverage. The thresholds and further documentation is contained in sections "R package covr" and "Testing steps".

The detailed coverage for the package is as follow:

| File | Coverage |
|---|---|
| R/apply_filters.R | 100.00% |
| R/bioavailability.R | 100.00% |
| R/calculate_summary_stats.R | 100.00% |
| R/conversions.R | 100.00% |
| R/create_start_impute.R | 100.00% |
| R/dose_profile_duplicates.R | 100.00% |
| R/exclude_nca_by_param.R | 100.00% |
| R/export_cdisc.R | 100.00% |
| R/filter_breaks.R | 78.79% |
| R/flexible_violinboxplot.R | 100.00% |
| R/format_data.R | 100.00% |
| R/g_pkcg.R | 100.00% |
| R/general_lineplot.R | 100.00% |
| R/general_meanplot.R | 100.00% |
| R/intervals_helpers.R | 100.00% |
| R/l_pkconc.R | 100.00% |
| R/label_operators.R | 100.00% |
| R/lambda_slope_plot.R | 100.00% |
| R/pivot_wider_pknca_results.R | 100.00% |
| R/PKNCA.R | 94.64% |
| R/ratio_calculations.R | 100.00% |
| R/read_pk.R | 100.00% |
| R/run_app.R | 0.00% |
| R/translate_terms.R | 100.00% |
| R/utils-slope_selector.R | 100.00% |
| R/utils.R | 100.00% |
| R/zzz.R | 0.00% |
| **Total coverage** | 90.42% |

### 4.7.5 Integration Testing using Reverse Dependency Checks

If the R package would update an existing version of the same package in the target environment, then integration testing of any reverse dependencies is as follows.

```
No issues identified.
```

### 4.7.6 Traceability Matrix

The traceability matrix traces tested behaviors back to which documented functionality is probed by the execution of the test.

**Requirements Naming Convention**

Requirement IDs correspond to subsections in *Section "Functional Requirements Specification"*.

The following table documents requirement IDs which have been evaluated by the corresponding test. Each row represents a single test case. A test, as denoted by a test ID and test name, may be comprised of multiple test cases, and may evaluate code related to multiple requirements.

| Req | Test ID | Test Name | Req Hits | Result |
|---|---|---|---|---|
| 4.4.1 | 24 | apply_filter throws error if column does not exist in data | 1 | PASSED |
| 4.4.1 | 25 | apply_filters skips NULL filters | 1 | PASSED |
| 4.4.1 | 26 | apply_filters throws error when filter misses required fields | 1 | PASSED |
| 4.4.1 | 27 | apply_filters works with != | 2 | PASSED |
| 4.4.1 | 28 | apply_filters works with compound filters | 1 | PASSED |
| 4.4.1 | 29 | apply_filters works with named filters | 1 | PASSED |
| 4.4.1 | 30 | apply_fitlers works with > | 4 | PASSED |
| 4.4.2 | 31 | `apply_labels(data, ADNCA_LABELS_FIXTURE, type = "ADPC")` | 1 | PASSED |
| 4.4.2 | 32 | apply_labels: does not change labels if already applied | 2 | PASSED |
| 4.4.2 | 33 | apply_labels: uses column names as labels when the variable is not in the labels list | 4 | PASSED |
| 4.4.3 | 31 | `apply_labels(data, ADNCA_LABELS_FIXTURE, type = "ADPC")` | 1 | PASSED |
| 4.4.3 | 32 | apply_labels: does not change labels if already applied | 2 | PASSED |
| 4.4.3 | 33 | apply_labels: uses column names as labels when the variable is not in the labels list | 4 | PASSED |
| 4.4.3 | 34 | as_factor_preserve_label: does not change the original label text | 1 | PASSED |
| 4.4.3 | 35 | `as_factor_preserve_label(mock_vec)` | 1 | PASSED |
| 4.4.4 | 36 | calculate_f: returns a data.frame with one column per bioavailability parameter | 2 | PASSED |
| 4.4.5 | 37 | calculate_summary_stats: executes efficiently on a large dataset | 1 | PASSED |
| 4.4.5 | 38 | calculate_summary_stats: handles an empty dataset | 1 | PASSED |
| 4.4.5 | 39 | calculate_summary_stats: returns a data frame | 1 | PASSED |
| 4.4.5 | 40 | calculate_summary_stats: standardizes units to the mode | 1 | PASSED |
| 4.4.5 | 41 | `calculate_summary_stats(test_data)` | 1 | PASSED |
| 4.4.6 | 11 | .filter_slopes: should handle slope exclusion | 2 | PASSED |
| 4.4.6 | 12 | .filter_slopes: should handle slope selection | 2 | PASSED |
| 4.4.6 | 42 | check_slope_rule_overlap: should add new points of partial overlap is detected | 1 | PASSED |

| Req | Test ID | Test Name | Req Hits | Result |
|---|---|---|---|---|
| 4.4.6 | 43 | check_slope_rule_overlap: should add new row if no overlap is detected | 3 | PASSED |
| 4.4.6 | 44 | check_slope_rule_overlap: should remove full row if full range of rule is removed | 1 | PASSED |
| 4.4.6 | 45 | check_slope_rule_overlap: should remove overlapping points if no new points are detected | 2 | PASSED |
| 4.4.6 | 46 | check_slope_rule_overlap: should warn if more than one range for single subject, profile and rule type is detected | 1 | PASSED |
| 4.4.7 | 247 | PKNCA_update_data_object: handles partial AUCs (auc_data) creating proper intervals for each | 2 | PASSED |
| 4.4.7 | 249 | PKNCA_update_data_object: returns a PKNCAdata object | 1 | PASSED |
| 4.4.7 | 250 | PKNCA_update_data_object: sets NCA options correctly | 3 | PASSED |
| 4.4.7 | 54 | create_start_impute: if drug column is not present, assumes is the same as analyte for imputation | 2 | PASSED |
| 4.4.7 | 55 | `create_start_impute(pknca_data)` | 1 | PASSED |
| 4.4.7 | 56 | `create_start_impute(PKNCA_DATA_FIXTURE)` | 1 | PASSED |
| 4.4.7 | 57 | create_start_impute: provides a warning when data$intervals is empty or has no rows | 1 | PASSED |
| 4.4.7 | 58 | create_start_impute: works without issue | 1 | PASSED |
| 4.4.8 | 113 | general_lineplot functions correctly: handles By Dose Profile time_scale with predose duplication | 1 | PASSED |
| 4.4.8 | 185 | lambda_slope_plot: handles NA in lambda.z.n.points gracefully | 1 | PASSED |
| 4.4.8 | 186 | lambda_slope_plot: returns a plotly object with valid input | 2 | PASSED |
| 4.4.8 | 187 | lambda_slope_plot: returns without error and gives expected warning when plot_data has 0 rows | 1 | PASSED |
| 4.4.8 | 188 | lambda_slope_plot: shows warning when Cmax is included in lambda estimation | 1 | PASSED |
| 4.4.8 | 189 | lambda_slope_plot: warns and returns empty plot when AVAL <= 0 | 2 | PASSED |
| 4.4.8 | 195 | pivot_wider_pknca_results: handles exclude values correctly | 2 | PASSED |
| 4.4.8 | 196 | `pivot_wider_pknca_results(pknca_res)` | 1 | PASSED |
| 4.4.8 | 197 | pivot_wider_pknca_results: produces a data.frame with expected format when only reshaping main intervals | 4 | PASSED |
| 4.4.8 | 36 | calculate_f: returns a data.frame with one column per bioavailability parameter | 2 | PASSED |
| 4.4.8 | 59 | `dose_profile_duplicates(conc_data, groups = c("USUBJID", "DOSNOA"), ; dosno = "DOSNOA")` | 1 | PASSED |
| 4.4.8 | 60 | dose_profile_duplicates: handles character or numeric dosno correctly | 1 | PASSED |
| 4.4.8 | 61 | dose_profile_duplicates: should return data with IX column if only one dose is present | 1 | PASSED |

| Req | Test ID | Test Name | Req Hits | Result |
|---|---|---|---|---|
| 4.4.9 | 62 | exclude_nca_by_param: does not exclude rows when max_thr is not exceeded | 1 | PASSED |
| 4.4.9 | 63 | exclude_nca_by_param: does not exclude rows when min_thr is not met | 1 | PASSED |
| 4.4.9 | 64 | exclude_nca_by_param: excludes rows based on max_thr | 1 | PASSED |
| 4.4.9 | 65 | exclude_nca_by_param: excludes rows based on min_thr | 1 | PASSED |
| 4.4.9 | 66 | exclude_nca_by_param: marks records associated with the affected_parameters | 1 | PASSED |
| 4.4.9 | 67 | exclude_nca_by_param: produces an error when more than 1 PPORRES is per parameter | 1 | PASSED |
| 4.4.9 | 68 | exclude_nca_by_param: returns the object when the parameter's value is NA | 1 | PASSED |
| 4.4.9 | 69 | exclude_nca_by_param: returns the original object when the parameter is not found | 1 | PASSED |
| 4.4.9 | 70 | exclude_nca_by_param: throws an error for invalid max_thr | 1 | PASSED |
| 4.4.9 | 71 | exclude_nca_by_param: throws an error for invalid min_thr | 1 | PASSED |
| 4.4.9 | 72 | exclude_nca_by_param: throws an error when min_thr is greater than max_thr | 1 | PASSED |
| 4.4.10 | 73 | export_cdisc: derives correctly PPRFTDTC (if either PCRFTDTC or PCRFTDTM are present) | 4 | PASSED |
| 4.4.10 | 74 | export_cdisc: derives PPGRPID correctly, using AVISIT, VISIT and/or PARAM, PCSPEC. NCA_PROFILE | 3 | PASSED |
| 4.4.10 | 75 | export_cdisc: derives PPREASND & PPSTAT correctly in all situations | 3 | PASSED |
| 4.4.10 | 76 | export_cdisc: derives PPSTINT and PPENINT for partial AUC intervals | 4 | PASSED |
| 4.4.10 | 77 | export_cdisc: derives when possible ATPT, ATPTN, ATPTREF (PCTPT, PCTPTNUM, PCTPTREF) | 6 | PASSED |
| 4.4.10 | 78 | export_cdisc: does not derive SUBJID if not present with USUBJID, STUDYID | 2 | PASSED |
| 4.4.10 | 79 | export_cdisc: exports CDISC-compliant datasets (PP, ADPP, ADPC) | 12 | PASSED |
| 4.4.11 | 108 | g_pkcg01_lin: generates plot with linear scale | 2 | PASSED |
| 4.4.11 | 109 | g_pkcg01_log: generates plot with log scale | 2 | PASSED |
| 4.4.11 | 110 | g_pkcg02_lin: generates plot with linear scale | 2 | PASSED |
| 4.4.11 | 111 | g_pkcg02_log: generates plot with log scale | 2 | PASSED |
| 4.4.11 | 198 | pkcg01: generates plotly plots with LIN scale | 2 | PASSED |
| 4.4.11 | 199 | pkcg01: generates plotly plots with LOG scale | 2 | PASSED |
| 4.4.11 | 200 | pkcg01: generates plotly plots with SBS scale | 2 | PASSED |
| 4.4.11 | 201 | pkcg01: generates plots with custom labels for LIN scale | 5 | PASSED |
| 4.4.11 | 202 | pkcg01: generates plots with custom multiple colors for LIN scale | 1 | PASSED |
| 4.4.11 | 203 | pkcg01: generates valid ggplots with LIN scale | 4 | PASSED |

| Req | Test ID | Test Name | Req Hits | Result |
|---|---|---|---|---|
| 4.4.11 | 204 | pkcg01: generates valid ggplots with LOG scale | 4 | PASSED |
| 4.4.11 | 205 | pkcg01: generates valid ggplots with SBS scale | 4 | PASSED |
| 4.4.11 | 206 | pkcg01: returns error if missing ggh4x package for SBS scale | 1 | PASSED |
| 4.4.11 | 207 | pkcg01: returns error if missing scales package for SBS scale | 1 | PASSED |
| 4.4.11 | 208 | pkcg02: generates plotly plots with LIN scale | 2 | PASSED |
| 4.4.11 | 209 | pkcg02: generates plotly plots with LOG scale | 2 | PASSED |
| 4.4.11 | 210 | pkcg02: generates plotly plots with SBS scale | 2 | PASSED |
| 4.4.11 | 211 | pkcg02: generates plots with custom groups and legend for LIN scale | 2 | PASSED |
| 4.4.11 | 212 | pkcg02: generates plots with custom labels for LIN scale | 5 | PASSED |
| 4.4.11 | 213 | pkcg02: generates valid ggplots with LIN scale | 3 | PASSED |
| 4.4.11 | 214 | pkcg02: generates valid ggplots with LOG scale | 3 | PASSED |
| 4.4.11 | 215 | pkcg02: generates valid ggplots with SBS scale | 3 | PASSED |
| 4.4.11 | 216 | pkcg02: returns error if missing ggh4x package for SBS scale | 1 | PASSED |
| 4.4.11 | 217 | pkcg02: returns error if missing scales package for SBS scale | 1 | PASSED |
| 4.4.12 | 11 | .filter_slopes: should handle slope exclusion | 2 | PASSED |
| 4.4.12 | 12 | .filter_slopes: should handle slope selection | 2 | PASSED |
| 4.4.12 | 13 | .filter_slopes: should return data unchanged if no slopes are provided | 2 | PASSED |
| 4.4.12 | 14 | .filter_slopes: should throw an error for invalid data | 4 | PASSED |
| 4.4.12 | 15 | .filter_slopes: should throw an error if reasons are missing | 1 | PASSED |
| 4.4.13 | 80 | flexible_violinboxplot: creates a plot with additional colorvars | 1 | PASSED |
| 4.4.13 | 81 | flexible_violinboxplot: creates a plot with additional varvalstofilter | 1 | PASSED |
| 4.4.13 | 82 | flexible_violinboxplot: creates a plot with additional xvars | 1 | PASSED |
| 4.4.13 | 83 | flexible_violinboxplot: creates a plotly object correctly | 1 | PASSED |
| 4.4.13 | 84 | flexible_violinboxplot: creates a simple plot with minimal arguments | 1 | PASSED |
| 4.4.13 | 85 | flexible_violinboxplot: creates a violin plot when box = FALSE | 1 | FAILED |
| 4.4.13 | 86 | flexible_violinboxplot: handles axis labels correctly when parameter has no unit | 2 | PASSED |
| 4.4.13 | 87 | flexible_violinboxplot: handles missing data gracefully | 1 | PASSED |
| 4.4.14 | 107 | format_pkncadose_data: handles negative time values | 1 | PASSED |
| 4.4.14 | 227 | PKNCA_calculate_nca: handles warning levels correctly | 1 | PASSED |
| 4.4.14 | 230 | `PKNCA_create_data_object(DUMMY_DATA_FIXTURE %>% filter(PCSPEC == ; "Plasma"))` | 1 | PASSED |

| Req | Test ID | Test Name | Req Hits | Result |
|---|---|---|---|---|
| 4.4.14 | 231 | PKNCA_create_data_object: handles duplicates in DFLAG | 1 | PASSED |
| 4.4.14 | 232 | PKNCA_create_data_object: handles missing columns required for PKNCA in the input data | 1 | PASSED |
| 4.4.14 | 233 | PKNCA_create_data_object: handles missing columns required for the functions in the input data | 2 | PASSED |
| 4.4.14 | 234 | `PKNCA_create_data_object(multiple_data)` | 1 | PASSED |
| 4.4.14 | 235 | PKNCA_create_data_object: produces a message error when missing values are in group columns | 1 | PASSED |
| 4.4.14 | 236 | `PKNCA_create_data_object(simple_data)` | 1 | PASSED |
| 4.4.14 | 88 | `format_pkncaconc_data(ADNCA, group_columns = c("STUDYID", "USUBJID", ; "PCSPEC", "DRUG", "PARAM"))` | 1 | PASSED |
| 4.4.14 | 89 | format_pkncaconc_data: filters EVID if column is present | 1 | PASSED |
| 4.4.14 | 90 | format_pkncaconc_data: filters out rows where PARAMCD contains DOSE | 1 | PASSED |
| 4.4.14 | 91 | format_pkncaconc_data: generates a valid dataset with required columns | 4 | PASSED |
| 4.4.14 | 92 | `format_pkncaconc_data(multi_analyte_adnca, group_columns = c("STUDYID", ; "USUBJID", "PCSPEC", "DRUG", "PARAM"), time_co` | 1 | PASSED |
| 4.4.14 | 93 | format_pkncaconc_data: processes multiple analytes correctly | 2 | PASSED |
| 4.4.14 | 94 | format_pkncaconc_data: returns an error for empty input dataframe | 1 | PASSED |
| 4.4.14 | 95 | format_pkncaconc_data: returns an error for missing required columns | 1 | PASSED |
| 4.4.15 | 100 | `format_pkncadata_intervals(PKNCA_DATA_FIXTURE$conc, PKNCA_DATA_FIXTURE$dose, ; params = c("aucinf.obs", "aucint.last", "` | 1 | PASSED |
| 4.4.15 | 101 | format_pkncadata_intervals: sets last time to end AFRLT if no TAU available | 1 | PASSED |
| 4.4.15 | 102 | format_pkncadata_intervals: uses ARRLT for start when start_from_last_dose is FALSE | 1 | PASSED |
| 4.4.15 | 246 | PKNCA_update_data_object: does not impute C0 when not requested | 2 | PASSED |
| 4.4.15 | 247 | PKNCA_update_data_object: handles partial AUCs (auc_data) creating proper intervals for each | 2 | PASSED |
| 4.4.15 | 248 | PKNCA_update_data_object: includes only selected analytes, dosnos, and pcspecs in intervals | 3 | PASSED |
| 4.4.15 | 249 | PKNCA_update_data_object: returns a PKNCAdata object | 1 | PASSED |
| 4.4.15 | 250 | PKNCA_update_data_object: sets NCA options correctly | 3 | PASSED |
| 4.4.15 | 96 | format_pkncadata_intervals: correctly uses tau if column is available | 1 | PASSED |
| 4.4.15 | 97 | format_pkncadata_intervals: handles incorrect input type | 2 | PASSED |
| 4.4.15 | 98 | format_pkncadata_intervals: handles missing columns | 1 | PASSED |

| Req | Test ID | Test Name | Req Hits | Result |
|---|---|---|---|---|
| 4.4.15 | 99 | format_pkncadata_intervals: handles multiple analytes with metabolites | 8 | PASSED |
| 4.4.16 | 103 | `format_pkncadose_data(df_conc, group_columns = c("STUDYID", "USUBJID", ; "PCSPEC", "DRUG"))` | 1 | PASSED |
| 4.4.16 | 104 | format_pkncadose_data: generates with no errors | 4 | PASSED |
| 4.4.16 | 105 | format_pkncadose_data: handles empty input | 1 | PASSED |
| 4.4.16 | 106 | format_pkncadose_data: handles missing columns | 1 | PASSED |
| 4.4.16 | 107 | format_pkncadose_data: handles negative time values | 1 | PASSED |
| 4.4.16 | 227 | PKNCA_calculate_nca: handles warning levels correctly | 1 | PASSED |
| 4.4.16 | 230 | `PKNCA_create_data_object(DUMMY_DATA_FIXTURE %>% filter(PCSPEC == ; "Plasma"))` | 1 | PASSED |
| 4.4.16 | 231 | PKNCA_create_data_object: handles duplicates in DFLAG | 1 | PASSED |
| 4.4.16 | 232 | PKNCA_create_data_object: handles missing columns required for PKNCA in the input data | 1 | PASSED |
| 4.4.16 | 234 | `PKNCA_create_data_object(multiple_data)` | 1 | PASSED |
| 4.4.16 | 236 | `PKNCA_create_data_object(simple_data)` | 1 | PASSED |
| 4.4.17 | 108 | g_pkcg01_lin: generates plot with linear scale | 2 | PASSED |
| 4.4.18 | 109 | g_pkcg01_log: generates plot with log scale | 2 | PASSED |
| 4.4.19 | 110 | g_pkcg02_lin: generates plot with linear scale | 2 | PASSED |
| 4.4.20 | 111 | g_pkcg02_log: generates plot with log scale | 2 | PASSED |
| 4.4.21 | 112 | general_lineplot functions correctly: can plot with logarithmic scale | 2 | PASSED |
| 4.4.21 | 113 | general_lineplot functions correctly: handles By Dose Profile time_scale with predose duplication | 1 | PASSED |
| 4.4.21 | 114 | general_lineplot functions correctly: handles empty data gracefully | 2 | PASSED |
| 4.4.21 | 115 | general_lineplot functions correctly: handles missing columns gracefully | 1 | PASSED |
| 4.4.21 | 116 | general_lineplot functions correctly: handles predose records when ARRLT < 0 and AFRLT > 0 | 2 | PASSED |
| 4.4.21 | 117 | general_lineplot functions correctly: handles time_scale not equal to 'By Cycle' | 2 | PASSED |
| 4.4.21 | 118 | general_lineplot functions correctly: removes non-positive AVAL values for log scale | 2 | PASSED |
| 4.4.21 | 119 | general_lineplot functions correctly: returns a ggplot object | 1 | PASSED |
| 4.4.21 | 120 | general_lineplot functions correctly: returns a ggplot object with no data available when preprocessed_data is empty | 2 | PASSED |
| 4.4.21 | 121 | general_lineplot functions correctly: supports multiple variables for colorby_var | 1 | PASSED |
| 4.4.22 | 122 | general_meanplot functions correctly: can plot with confidence interval ribbon | 3 | PASSED |

| Req | Test ID | Test Name | Req Hits | Result |
|---|---|---|---|---|
| 4.4.22 | 123 | general_meanplot functions correctly: can plot with logarithmic scale | 2 | PASSED |
| 4.4.22 | 124 | general_meanplot functions correctly: can plot with standard deviation error bars | 3 | PASSED |
| 4.4.22 | 125 | general_meanplot functions correctly: handles empty data gracefully | 3 | PASSED |
| 4.4.22 | 126 | general_meanplot functions correctly: handles missing columns gracefully | 1 | PASSED |
| 4.4.22 | 127 | general_meanplot functions correctly: returns a ggplot object | 2 | PASSED |
| 4.4.24 | 134 | get_label: returns label of a heading if it exists in the label file | 1 | PASSED |
| 4.4.24 | 135 | get_label: returns 'No label available' if the label does not exist | 1 | PASSED |
| 4.4.25 | 136 | `has_label(mock_vec)` | 1 | PASSED |
| 4.4.25 | 137 | has_label: returns FALSE if has no label | 1 | PASSED |
| 4.4.25 | 138 | has_label: returns TRUE if has label | 1 | PASSED |
| 4.4.26 | 139 | interval_add_impute: adds new rows with added imputations after the original ones | 1 | PASSED |
| 4.4.26 | 140 | interval_add_impute: adds the impute method in the impute column of a dummy intervals dataframe | 1 | PASSED |
| 4.4.26 | 141 | interval_add_impute and interval_remove_impute: are inverses of each other | 1 | PASSED |
| 4.4.26 | 142 | interval_add_impute: creates missing impute col as NA_char & adds impute | 2 | PASSED |
| 4.4.26 | 143 | interval_add_impute: does not add new interval if non-target & target params share target impute | 1 | PASSED |
| 4.4.26 | 144 | interval_add_impute: does not create duplicates but removes the originals & adds impute method based on after | 1 | PASSED |
| 4.4.26 | 145 | interval_add_impute: handles impute column with FALSE values correctly | 1 | PASSED |
| 4.4.26 | 146 | interval_add_impute: handles mixed TRUE/FALSE for cmax and half.life correctly | 1 | PASSED |
| 4.4.26 | 147 | interval_add_impute: handles multiple target_params correctly | 1 | PASSED |
| 4.4.26 | 148 | interval_add_impute: handles specified target_params correctly | 2 | PASSED |
| 4.4.26 | 149 | interval_add_impute: handles target_groups correctly | 2 | PASSED |
| 4.4.26 | 151 | interval_add_impute: reports an error when the impute column is not a character | 1 | PASSED |
| 4.4.26 | 152 | interval_add_impute: throws an error for non-character target_impute | 1 | PASSED |
| 4.4.26 | 153 | interval_add_impute: throws an error for unknown target_params | 1 | PASSED |

| Req | Test ID | Test Name | Req Hits | Result |
|---|---|---|---|---|
| 4.4.26 | 154 | interval_add_impute: throws an error if either data or target_impute is missing | 1 | PASSED |
| 4.4.26 | 155 | interval_add_impute: throws an error when input data is not a proper format object | 2 | PASSED |
| 4.4.26 | 156 | interval_add_impute: warns and makes no changes when target_impute is NA or empty | 4 | PASSED |
| 4.4.26 | 157 | interval_add_impute: with no optional parameters uses all, with new intervals below | 1 | PASSED |
| 4.4.27 | 141 | interval_add_impute and interval_remove_impute: are inverses of each other | 1 | PASSED |
| 4.4.27 | 150 | interval_add_impute: makes no changes and warns when no matching intervals are found | 2 | PASSED |
| 4.4.27 | 158 | interval_remove_impute: does not modify data if global impute & column are missing | 4 | PASSED |
| 4.4.27 | 159 | interval_remove_impute: handles impute column with FALSE values correctly | 1 | PASSED |
| 4.4.27 | 160 | interval_remove_impute: handles mixed TRUE/FALSE for cmax and half.life correctly | 1 | PASSED |
| 4.4.27 | 161 | interval_remove_impute: handles multiple target_params correctly | 1 | PASSED |
| 4.4.27 | 162 | interval_remove_impute: handles properly impute character method with multiple imputes | 1 | PASSED |
| 4.4.27 | 163 | interval_remove_impute: handles specified target_params correctly | 2 | PASSED |
| 4.4.27 | 164 | interval_remove_impute: handles target_groups correctly | 2 | PASSED |
| 4.4.27 | 165 | interval_remove_impute: if impute col is missing uses global impute | 3 | PASSED |
| 4.4.27 | 166 | interval_remove_impute: includes new rows right after the original ones | 1 | PASSED |
| 4.4.27 | 167 | interval_remove_impute: makes no changes and warns when no matching intervals found | 2 | PASSED |
| 4.4.27 | 168 | interval_remove_impute: removes all target_impute even if is several times | 1 | PASSED |
| 4.4.27 | 169 | interval_remove_impute: removes the impute method in the impute column of a dummy intervals dataframe | 1 | PASSED |
| 4.4.27 | 170 | interval_remove_impute: reports an error when impute column is not a character | 1 | PASSED |
| 4.4.27 | 171 | interval_remove_impute: throws an error for non-character target_impute | 1 | PASSED |
| 4.4.27 | 172 | interval_remove_impute: throws an error for unknown target_params | 1 | PASSED |
| 4.4.27 | 173 | interval_remove_impute: throws an error if either data or target_impute is missing | 1 | PASSED |
| 4.4.27 | 174 | interval_remove_impute: throws an error when input data is not in correct format | 2 | PASSED |

| Req | Test ID | Test Name | Req Hits | Result |
|---|---|---|---|---|
| 4.4.27 | 175 | interval_remove_impute: warns and makes no changes when target_impute is NA or empty | 4 | PASSED |
| 4.4.27 | 176 | interval_remove_impute: with no optional parameters uses all relevant cases | 1 | PASSED |
| 4.4.27 | 247 | PKNCA_update_data_object: handles partial AUCs (auc_data) creating proper intervals for each | 2 | PASSED |
| 4.4.27 | 249 | PKNCA_update_data_object: returns a PKNCAdata object | 1 | PASSED |
| 4.4.27 | 250 | PKNCA_update_data_object: sets NCA options correctly | 3 | PASSED |
| 4.4.28 | 177 | l_pkconc: creates listings for each unique combination of grouping variables | 12 | PASSED |
| 4.4.28 | 178 | l_pkconc: handles custom formatting_vars_table | 27 | PASSED |
| 4.4.28 | 179 | l_pkconc: handles empty data frame by providing empty list | 1 | PASSED |
| 4.4.28 | 180 | l_pkconc: handles missing footnote (no footnote) | 1 | PASSED |
| 4.4.28 | 181 | l_pkconc: handles missing formatting_vars_table and uses a default built: | 1 | PASSED |
| 4.4.28 | 182 | l_pkconc: handles missing required columns | 1 | PASSED |
| 4.4.28 | 183 | l_pkconc: handles missing subtitle and creates a default | 1 | PASSED |
| 4.4.28 | 184 | l_pkconc: handles non-unique units | 1 | PASSED |
| 4.4.29 | 185 | lambda_slope_plot: handles NA in lambda.z.n.points gracefully | 1 | PASSED |
| 4.4.29 | 186 | lambda_slope_plot: returns a plotly object with valid input | 2 | PASSED |
| 4.4.29 | 187 | lambda_slope_plot: returns without error and gives expected warning when plot_data has 0 rows | 1 | PASSED |
| 4.4.29 | 188 | lambda_slope_plot: shows warning when Cmax is included in lambda estimation | 1 | PASSED |
| 4.4.29 | 189 | lambda_slope_plot: warns and returns empty plot when AVAL <= 0 | 2 | PASSED |
| 4.4.30 | 190 | multiple_matrix_ratios function: computes correct ratios | 1 | PASSED |
| 4.4.30 | 191 | multiple_matrix_ratios function: handles missing data correctly | 1 | PASSED |
| 4.4.30 | 192 | multiple_matrix_ratios function: handles non-matching time points correctly | 1 | PASSED |
| 4.4.31 | 108 | g_pkcg01_lin: generates plot with linear scale | 2 | PASSED |
| 4.4.31 | 109 | g_pkcg01_log: generates plot with log scale | 2 | PASSED |
| 4.4.31 | 110 | g_pkcg02_lin: generates plot with linear scale | 2 | PASSED |
| 4.4.31 | 111 | g_pkcg02_log: generates plot with log scale | 2 | PASSED |
| 4.4.31 | 177 | l_pkconc: creates listings for each unique combination of grouping variables | 12 | PASSED |
| 4.4.31 | 178 | l_pkconc: handles custom formatting_vars_table | 27 | PASSED |
| 4.4.31 | 179 | l_pkconc: handles empty data frame by providing empty list | 1 | PASSED |
| 4.4.31 | 180 | l_pkconc: handles missing footnote (no footnote) | 1 | PASSED |

| Req | Test ID | Test Name | Req Hits | Result |
|---|---|---|---|---|
| 4.4.31 | 181 | l_pkconc: handles missing formatting_vars_table and uses a default built: | 1 | PASSED |
| 4.4.31 | 183 | l_pkconc: handles missing subtitle and creates a default | 1 | PASSED |
| 4.4.31 | 184 | l_pkconc: handles non-unique units | 1 | PASSED |
| 4.4.31 | 193 | parse_annotation: parses title string correctly | 1 | PASSED |
| 4.4.31 | 194 | parse_annotation: substitutes missing variables with ERR | 2 | PASSED |
| 4.4.31 | 198 | pkcg01: generates plotly plots with LIN scale | 2 | PASSED |
| 4.4.31 | 199 | pkcg01: generates plotly plots with LOG scale | 2 | PASSED |
| 4.4.31 | 200 | pkcg01: generates plotly plots with SBS scale | 2 | PASSED |
| 4.4.31 | 201 | pkcg01: generates plots with custom labels for LIN scale | 5 | PASSED |
| 4.4.31 | 202 | pkcg01: generates plots with custom multiple colors for LIN scale | 1 | PASSED |
| 4.4.31 | 203 | pkcg01: generates valid ggplots with LIN scale | 4 | PASSED |
| 4.4.31 | 204 | pkcg01: generates valid ggplots with LOG scale | 4 | PASSED |
| 4.4.31 | 205 | pkcg01: generates valid ggplots with SBS scale | 4 | PASSED |
| 4.4.31 | 206 | pkcg01: returns error if missing ggh4x package for SBS scale | 1 | PASSED |
| 4.4.31 | 207 | pkcg01: returns error if missing scales package for SBS scale | 1 | PASSED |
| 4.4.31 | 208 | pkcg02: generates plotly plots with LIN scale | 2 | PASSED |
| 4.4.31 | 209 | pkcg02: generates plotly plots with LOG scale | 2 | PASSED |
| 4.4.31 | 210 | pkcg02: generates plotly plots with SBS scale | 2 | PASSED |
| 4.4.31 | 211 | pkcg02: generates plots with custom groups and legend for LIN scale | 2 | PASSED |
| 4.4.31 | 212 | pkcg02: generates plots with custom labels for LIN scale | 5 | PASSED |
| 4.4.31 | 213 | pkcg02: generates valid ggplots with LIN scale | 3 | PASSED |
| 4.4.31 | 214 | pkcg02: generates valid ggplots with LOG scale | 3 | PASSED |
| 4.4.31 | 215 | pkcg02: generates valid ggplots with SBS scale | 3 | PASSED |
| 4.4.31 | 216 | pkcg02: returns error if missing ggh4x package for SBS scale | 1 | PASSED |
| 4.4.31 | 217 | pkcg02: returns error if missing scales package for SBS scale | 1 | PASSED |
| 4.4.32 | 195 | pivot_wider_pknca_results: handles exclude values correctly | 2 | PASSED |
| 4.4.32 | 196 | `pivot_wider_pknca_results(pknca_res)` | 1 | PASSED |
| 4.4.32 | 197 | pivot_wider_pknca_results: produces a data.frame with expected format when only reshaping main intervals | 4 | PASSED |
| 4.4.32 | 36 | calculate_f: returns a data.frame with one column per bioavailability parameter | 2 | PASSED |
| 4.4.33 | 108 | g_pkcg01_lin: generates plot with linear scale | 2 | PASSED |
| 4.4.33 | 109 | g_pkcg01_log: generates plot with log scale | 2 | PASSED |
| 4.4.33 | 198 | pkcg01: generates plotly plots with LIN scale | 2 | PASSED |

| Req | Test ID | Test Name | Req Hits | Result |
|---|---|---|---|---|
| 4.4.33 | 199 | pkcg01: generates plotly plots with LOG scale | 2 | PASSED |
| 4.4.33 | 200 | pkcg01: generates plotly plots with SBS scale | 2 | PASSED |
| 4.4.33 | 201 | pkcg01: generates plots with custom labels for LIN scale | 5 | PASSED |
| 4.4.33 | 202 | pkcg01: generates plots with custom multiple colors for LIN scale | 1 | PASSED |
| 4.4.33 | 203 | pkcg01: generates valid ggplots with LIN scale | 4 | PASSED |
| 4.4.33 | 204 | pkcg01: generates valid ggplots with LOG scale | 4 | PASSED |
| 4.4.33 | 205 | pkcg01: generates valid ggplots with SBS scale | 4 | PASSED |
| 4.4.33 | 206 | pkcg01: returns error if missing ggh4x package for SBS scale | 1 | PASSED |
| 4.4.33 | 207 | pkcg01: returns error if missing scales package for SBS scale | 1 | PASSED |
| 4.4.34 | 110 | g_pkcg02_lin: generates plot with linear scale | 2 | PASSED |
| 4.4.34 | 111 | g_pkcg02_log: generates plot with log scale | 2 | PASSED |
| 4.4.34 | 208 | pkcg02: generates plotly plots with LIN scale | 2 | PASSED |
| 4.4.34 | 209 | pkcg02: generates plotly plots with LOG scale | 2 | PASSED |
| 4.4.34 | 210 | pkcg02: generates plotly plots with SBS scale | 2 | PASSED |
| 4.4.34 | 211 | pkcg02: generates plots with custom groups and legend for LIN scale | 2 | PASSED |
| 4.4.34 | 212 | pkcg02: generates plots with custom labels for LIN scale | 5 | PASSED |
| 4.4.34 | 213 | pkcg02: generates valid ggplots with LIN scale | 3 | PASSED |
| 4.4.34 | 214 | pkcg02: generates valid ggplots with LOG scale | 3 | PASSED |
| 4.4.34 | 215 | pkcg02: generates valid ggplots with SBS scale | 3 | PASSED |
| 4.4.34 | 216 | pkcg02: returns error if missing ggh4x package for SBS scale | 1 | PASSED |
| 4.4.34 | 217 | pkcg02: returns error if missing scales package for SBS scale | 1 | PASSED |
| 4.4.35 | 218 | PKNCA_build_units_table: creates a NA units tables when units are not defined in the PKNCA objects | 4 | PASSED |
| 4.4.35 | 219 | PKNCA_build_units_table: creates an uniform units table when units are not defined as columns in the PKNCA obj | 4 | PASSED |
| 4.4.35 | 220 | `PKNCA_build_units_table(o_conc, o_dose)` | 1 | PASSED |
| 4.4.35 | 221 | PKNCA_build_units_table: reports an error when units are not uniform through all concentration groups | 1 | PASSED |
| 4.4.35 | 227 | PKNCA_calculate_nca: handles warning levels correctly | 1 | PASSED |
| 4.4.35 | 230 | `PKNCA_create_data_object(DUMMY_DATA_FIXTURE %>% filter(PCSPEC == ; "Plasma"))` | 1 | PASSED |
| 4.4.35 | 231 | PKNCA_create_data_object: handles duplicates in DFLAG | 1 | PASSED |
| 4.4.35 | 234 | `PKNCA_create_data_object(multiple_data)` | 1 | PASSED |
| 4.4.35 | 236 | `PKNCA_create_data_object(simple_data)` | 1 | PASSED |
| 4.4.36 | 22 | add_f_to_pknca_results: adds bioavailability parameters to the pknca result | 4 | PASSED |

| Req | Test ID | Test Name | Req Hits | Result |
|---|---|---|---|---|
| 4.4.36 | 222 | pknca_calculate_f: calculates ratios only when start and end match | 2 | PASSED |
| 4.4.36 | 223 | `pknca_calculate_f(pknca_res, c("f_AUCLST"))` | 1 | PASSED |
| 4.4.36 | 224 | pknca_calculate_f: returns NULL for missing or unavailable AUCs, last case with a warning | 3 | PASSED |
| 4.4.36 | 225 | pknca_calculate_f: uses intravascular and extravascular data for calculations when available | 1 | PASSED |
| 4.4.36 | 226 | pknca_calculate_f: when needed handles unit conversions if possible for the ratio calculations | 1 | PASSED |
| 4.4.36 | 36 | calculate_f: returns a data.frame with one column per bioavailability parameter | 2 | PASSED |
| 4.4.37 | 228 | `PKNCA_calculate_nca(pknca_data)` | 1 | PASSED |
| 4.4.37 | 229 | `PKNCA_calculate_nca(PKNCA_DATA_FIXTURE)` | 1 | PASSED |
| 4.4.38 | 227 | PKNCA_calculate_nca: handles warning levels correctly | 1 | PASSED |
| 4.4.38 | 230 | `PKNCA_create_data_object(DUMMY_DATA_FIXTURE %>% filter(PCSPEC == ; "Plasma"))` | 1 | PASSED |
| 4.4.38 | 231 | PKNCA_create_data_object: handles duplicates in DFLAG | 1 | PASSED |
| 4.4.38 | 232 | PKNCA_create_data_object: handles missing columns required for PKNCA in the input data | 1 | PASSED |
| 4.4.38 | 233 | PKNCA_create_data_object: handles missing columns required for the functions in the input data | 2 | PASSED |
| 4.4.38 | 234 | `PKNCA_create_data_object(multiple_data)` | 1 | PASSED |
| 4.4.38 | 235 | PKNCA_create_data_object: produces a message error when missing values are in group columns | 1 | PASSED |
| 4.4.38 | 236 | `PKNCA_create_data_object(simple_data)` | 1 | PASSED |
| 4.4.39 | 229 | `PKNCA_calculate_nca(PKNCA_DATA_FIXTURE)` | 1 | PASSED |
| 4.4.39 | 237 | PKNCA_impute_method_start_c1: does not impute when start is in the data | 1 | PASSED |
| 4.4.39 | 238 | PKNCA_impute_method_start_c1: ignores data outside the interval (before interval) | 1 | PASSED |
| 4.4.39 | 239 | PKNCA_impute_method_start_c1: imputes when start is not in the data | 1 | PASSED |
| 4.4.39 | 240 | `PKNCA_impute_method_start_logslope(conc = c(5, 4, 3, 2, 1), time = c(0.5, ; 1.5, 2.5, 3.5, 4.5), start = 0, end = 5, con` | 1 | PASSED |
| 4.4.40 | 229 | `PKNCA_calculate_nca(PKNCA_DATA_FIXTURE)` | 1 | PASSED |
| 4.4.40 | 240 | `PKNCA_impute_method_start_logslope(conc = c(5, 4, 3, 2, 1), time = c(0.5, ; 1.5, 2.5, 3.5, 4.5), start = 0, end = 5, con` | 1 | PASSED |
| 4.4.40 | 241 | PKNCA_impute_method_start_logslope: does not impute when start is in the data | 1 | PASSED |
| 4.4.40 | 242 | PKNCA_impute_method_start_logslope: does not modify if C1 = C2 in samples | 1 | PASSED |

| Req | Test ID | Test Name | Req Hits | Result |
|---|---|---|---|---|
| 4.4.40 | 243 | PKNCA_impute_method_start_logslope: does not modify if no C1 -> C2 decline in samples | 1 | PASSED |
| 4.4.40 | 244 | PKNCA_impute_method_start_logslope: ignores data outside the interval (before interval) | 1 | PASSED |
| 4.4.40 | 245 | PKNCA_impute_method_start_logslope: imputes when start is not in the data | 1 | PASSED |
| 4.4.41 | 246 | PKNCA_update_data_object: does not impute C0 when not requested | 2 | PASSED |
| 4.4.41 | 247 | PKNCA_update_data_object: handles partial AUCs (auc_data) creating proper intervals for each | 2 | PASSED |
| 4.4.41 | 248 | PKNCA_update_data_object: includes only selected analytes, dosnos, and pcspecs in intervals | 3 | PASSED |
| 4.4.41 | 249 | PKNCA_update_data_object: returns a PKNCAdata object | 1 | PASSED |
| 4.4.41 | 250 | PKNCA_update_data_object: sets NCA options correctly | 3 | PASSED |
| 4.4.42 | 251 | read_pk: reads csv data correctly | 2 | PASSED |
| 4.4.42 | 252 | read_pk: reads excel data correctly | 2 | PASSED |
| 4.4.42 | 253 | read_pk: reads parquet files correctly | 2 | PASSED |
| 4.4.42 | 254 | read_pk: reads rds data correctly | 2 | PASSED |
| 4.4.42 | 255 | read_pk: reads sas data correctly | 2 | PASSED |
| 4.4.42 | 256 | read_pk: reads xpt files correctly | 2 | PASSED |
| 4.4.42 | 257 | read_pk: throws an error if file does not exist | 1 | PASSED |
| 4.4.42 | 258 | read_pk: throws an error if file with unsupported format is loaded | 1 | PASSED |
| 4.4.42 | 259 | read_pk: throws an error if loaded data frame has no rows | 1 | PASSED |
| 4.4.42 | 260 | read_pk: throws an error if loaded object is not a data frame | 1 | PASSED |
| 4.4.44 | 265 | set_empty_label: sets label to empty string if it does not exist | 1 | PASSED |
| 4.4.45 | 195 | pivot_wider_pknca_results: handles exclude values correctly | 2 | PASSED |
| 4.4.45 | 196 | `pivot_wider_pknca_results(pknca_res)` | 1 | PASSED |
| 4.4.45 | 197 | pivot_wider_pknca_results: produces a data.frame with expected format when only reshaping main intervals | 4 | PASSED |
| 4.4.45 | 266 | translate_terms: handles invalid mapping_col and target_col types | 2 | PASSED |
| 4.4.45 | 267 | translate_terms: handles missing mapping_col gracefully | 1 | PASSED |
| 4.4.45 | 268 | translate_terms: handles missing metadata using the default from pknca_cdisc_terms.rds | 1 | PASSED |
| 4.4.45 | 269 | translate_terms: handles missing target_col gracefully | 1 | PASSED |
| 4.4.45 | 270 | `translate_terms(PPTESTCD, "PKNCA", "PPTESTCD")` | 1 | PASSED |
| 4.4.45 | 271 | `translate_terms(PPTESTCD, "PPTESTCD", "PKNCA")` | 1 | PASSED |
| 4.4.45 | 272 | translate_terms: translates terms correctly | 1 | PASSED |

| Req | Test ID | Test Name | Req Hits | Result |
|---|---|---|---|---|
| 4.4.45 | 36 | calculate_f: returns a data.frame with one column per bioavailability parameter | 2 | PASSED |
| 4.4.45 | 62 | exclude_nca_by_param: does not exclude rows when max_thr is not exceeded | 1 | PASSED |
| 4.4.45 | 63 | exclude_nca_by_param: does not exclude rows when min_thr is not met | 1 | PASSED |
| 4.4.45 | 64 | exclude_nca_by_param: excludes rows based on max_thr | 1 | PASSED |
| 4.4.45 | 65 | exclude_nca_by_param: excludes rows based on min_thr | 1 | PASSED |
| 4.4.45 | 66 | exclude_nca_by_param: marks records associated with the affected_parameters | 1 | PASSED |
| 4.4.45 | 73 | export_cdisc: derives correctly PPRFTDTC (if either PCRFTDTC or PCRFTDTM are present) | 4 | PASSED |
| 4.4.45 | 74 | export_cdisc: derives PPGRPID correctly, using AVISIT, VISIT and/or PARAM, PCSPEC. NCA_PROFILE | 3 | PASSED |
| 4.4.45 | 75 | export_cdisc: derives PPREASND & PPSTAT correctly in all situations | 3 | PASSED |
| 4.4.45 | 76 | export_cdisc: derives PPSTINT and PPENINT for partial AUC intervals | 4 | PASSED |
| 4.4.45 | 77 | export_cdisc: derives when possible ATPT, ATPTN, ATPTREF (PCTPT, PCTPTNUM, PCTPTREF) | 6 | PASSED |
| 4.4.45 | 78 | export_cdisc: does not derive SUBJID if not present with USUBJID, STUDYID | 2 | PASSED |
| 4.4.45 | 79 | export_cdisc: exports CDISC-compliant datasets (PP, ADPP, ADPC) | 12 | PASSED |

### 4.7.7 Autovalidate Execution Log

The following output was generated by the `autovalidate` tool before compiling the report.

```
                                    _ _     _       _
  ___                      __   ____ _| (_) __| | __ _| |_ ___
    _-_-  _/\______\__      \ \ / / _` | | |/ _` |/ _` | __/ _ \
 _-_-__  / ,-. -|-  ,-.`-.   \ V / (_| | | | (_| | (_| | ||  __/
    _-_- `( o )----( o )-'    \_/ \__,_|_|_|\__,_|\__,_|\__\___|
          `-'      `-'

.-~-.-~-.-~-.-~-.-~-.-~-.-~-.-~-.-~-.-~-.-~-.-~-.-~-.-~-.-~-.-~-.-~-.-~-.-~

  version: 2.15.0



WARNING /builds/autovalidate-r/package-
submission/portal/validation/validation_report.md

Consistency checks failed:

 - requirement #0.23 'Transform Units' is not covered by any test
```

```
- requirement #0.43 'Run the Shiny app' is not covered by any test
- the test 'flexible_violinboxplot_creates_a_violin_plot_when_box_FALSE' has
  non-passing status, 'FAILED'
- [autovalidateutils] ERROR: checking tests ...
    Running ‘testthat.R’
   ERROR
  Running the tests in ‘tests/testthat.R’ failed.
  Last 13 lines of output:
    ── Failure ('test-flexible_violinboxplot.R:78:5'):
  flexible_violinboxplot: creates a violin plot when box = FALSE ──
    Snapshot of `testcase` to 'flexible_violinboxplot/violin-plot.svg' has
  changed
    * Download and unzip run artifact
    * Copy 'tests/testthat/_snaps/flexible_violinboxplot/violin-
  plot.new.svg' to local test directory
    * Run `testthat::snapshot_review('flexible_violinboxplot/')` to review
  changes
    Backtrace:
        ▆
     1. └─vdiffr::expect_doppelganger("violin_plot", violin_plot) at test-
  flexible_violinboxplot.R:78:5
     2.   ├─base::withCallingHandlers(...)
     3.   └─testthat::expect_snapshot_file(...)

    [ FAIL 1 | WARN 0 | SKIP 0 | PASS 547 ]
    Error: Test failures
    In addition: There were 12 warnings (use warnings() to see them)
    Execution halted
  Running the tests in ‘tests/vdiffr.[rR]’ failed.
  Last 13 lines of output:
      'spacingAndGlyphs'>CMAX  [ ng/mL ]</text>
      <rect x='627.13' y='253.52' width='87.39' height='60.85'
  style='stroke-width:
      1.07; stroke: none; fill: #FFFFFF;' />
    @@ 69,5 / 72,5 @@
      <text x='655.37' y='286.00' style='font-size: 8.80px; font-family:
  sans;' text
      Length='4.89px' lengthAdjust='spacingAndGlyphs'>1</text>
      <text x='655.37' y='303.28' style='font-size: 8.80px; font-family:
  sans;' text
      Length='4.89px' lengthAdjust='spacingAndGlyphs'>2</text>
    < <text x='32.79' y='14.56' style='font-size: 13.20px; font-family:
  sans;' textL
    : ength='58.70px' lengthAdjust='spacingAndGlyphs'>violin_plot</text>
    > <text x='27.90' y='14.56' style='font-size: 13.20px; font-family:
  sans;' textL
    : ength='58.70px' lengthAdjust='spacingAndGlyphs'>violin_plot</text>
      </g>
      </svg>


OK /builds/autovalidate-r/package-
submission/portal/validation/validation_report.md
```

## 4.8 Remediation

If the validation of the R package encountered any conditions which require remediation, as evident through warnings itemized in the execution of `autovalidate`, then justification to permit the package to be used despite the identified gaps is included below.

### 4.8.1 Remediation Comments

**Lorenzo Braschi** (@braschil) at 2025-06-24T15:02:52.118+02:00

#### 4.8.1.1 Missing requirements

The `aNCA` package is well covered and comes close to pass automatic validation. Two missing requirements are flagged: 'Transform Units' and 'Run Shiny App'. Upon inspection, the first one is an error on our side, as the function defined inside a `Vectorize()` function cannot be easily mapped onto the traceability matrix. Still, the tests run and pass, so we can consider this issue closed. The missing test for `Run Shiny App' is also ignorable (and in any case such tests exist, they are just excluded from`covr') as the core functionality that makes the Shiny app is tested with other tests.

#### 4.8.1.2 Failing snapshot test

The failing test for the violin plot means that the output of the generated plot does not match exactly the version stored as reference. Upon inspection, these differences are deemed ignorable and do not pose a threat to validation.

In light of the above, we can accept `aNCA` for validation.

Approved by **Lorenzo Braschi** (@braschil)

# 5 Validation Summary

## 5.1 Validation Activities Results

- All validation activities defined within the Activities Sequence section have been successfully completed.
- All evaluation and available testing activities were successfully completed as described in the Testing Plan.
- The Testing Summary contains test results from the automated execution of installation and any available unit tests. This documentation, which proves that the package can be successfully installed within the testing environment.

## 5.2 Conclusions

All available test cases for the R package were successfully executed in the testing environment and all validation activities were completed according to Validation Plan. All business requirements and IT procedures as stated in the *Validation Plan* have been satisfied. Based on the reviewed R package metadata and test results, it has been demonstrated that the R Package "aNCA" version 0.0.0.9095 is deemed validated and acceptable for release to production use on Target (Deployment) Environment.

The Target Environment must reflect the Testing Environment as described in Section *Terms and Abbreviations*. The scope of validated use on those platforms extends only as far as the systems qualification of the target system. Specifically, R packages may only be used for regulated outputs generation within systems qualified for that purpose.

## 5.3 Validation Registry

This section provides the list of document deliverables as described in the Validation Plan and Validation Report sections. The Validation Registry identifies and references all valid and effective documents associated with the system, just prior to go-live.

1. `autovalidate` System Validation (SAP ID: `20271310`).
2. Software Validation using Autovalidation System Risk Assessment (SYS ID: SYS-5762, SRA-22510) located in iRAAM+.
3. Validation Report located in the Veeva Quality Docs (ID Project: R Package Validation).
4. Operational Support Plan (QD ID: `24882646`).
5. Cornerstone Training (Cornerstone Training: Autovalidate R Training).

# 6 Appendix

## 6.1 Environment variables

| Key | Value |
|---|---|
| AUTOVALIDATEUTILS_BUILD_BINARIES | TRUE |
| AUTOVALIDATEUTILS_GENERALIZE_SOURCES | TRUE |
| AUTOVALIDATEUTILS_INST_DEPS | TRUE |
| AUTOVALIDATEUTILS_REVDEP_INTERNAL_TIMEOUT | as.difftime(240,units=”mins”) |
| AUTOVALIDATEUTILS_R_CMD_CHECK_VARIABLES | internal |
| AUTOVALIDATEUTILS_STEPS | '-write_revdep_check' |
| AUTOVALIDATEUTILS_SUGGESTS_REPOS | https://rspm.roche.com/CRAN/__linux__/jammy/2025-05-23 |
| AUTOVALIDATEUTILS_TIMEOUT | as.difftime(16,units=”hours”) |
| AUTOVALIDATEUTILS_UPGRADE_DEPS | TRUE |
| AUTOVALIDATEUTILS_USE_TESTTHAT | TRUE |
| AUTOVALIDATE_IMAGE_ID | registry.code.roche.com/autovalidate/autovalidate:2.15.0 |
| AUTOVALIDATE_IMAGE_TAG | 2.15.0 |
| AUTOVALIDATE_IS_AUTOMATED_AUTOVALIDATE_R_PIPELINE | true |
| AUTOVALIDATE_LEGACY_MERGE_REQUEST_HANDLER_USER | project_121925_bot1;project_21640_bot1;ghost |
| AUTOVALIDATE_MERGE_REQUEST_HANDLER_USER | group_14649_bot_eba51c89917b168fe87f0a6476ef9b2e |
| AUTOVALIDATE_QD_API_URL | https://prod-de-c1.apis.roche.com:443/pharma-qdocs-doc-upd-proc/v1/qualitydocumentupload |
| AUTOVALIDATE_QD_API_USER | 84181f50e7614d76baf40a17fea219aa |
| AUTOVALIDATE_QD_APPROVERS | {”maksymis”:”user:12095143”,”braschil”:”user:15033487”,”kelkhofd”:”user:11931829”} |

| Key | Value |
|---|---|
| AUTOVALIDATE_QD_APPROVER_API_URL | https://prod-de-c1.apis.roche.com:443/pharma-qdocs-doc-upd-proc/v1/workflow/Objectworkflow.d2a_approval__c |
| AUTOVALIDATE_R | true |
| AUTOVALIDATE_ROCHEPDF_IMAGE_TAG | v2.2.1 |
| AUTOVALIDATE_R_EARLY_TERMINATION_MSG | |
| AUTOVALIDATE_R_IMAGE | |
| AUTOVALIDATE_R_IMAGE_PRODUCT_NAME | r_minimal |
| AUTOVALIDATE_R_IMAGE_R_VERSION | 4.4.3 |
| AUTOVALIDATE_R_PORTAL_VARIABLE_FILTER_EXPRESSION | /AUTOVALIDATEUTILS_.*/d;/AUTOVALIDATE_.*/d;/_R.*/d;/CI_.*/d;/PORTAL_.*/d;/^[[:space:]]*$/d |
| AUTOVALIDATE_R_REPORT_BRANCH | master |
| AUTOVALIDATE_R_REPORT_REPO | autovalidate-r/package-report-template |
| AUTOVALIDATE_R_REPORT_SUBDIR | . |
| AUTOVALIDATE_R_SOURCE | https://github.com/pharmaverse/aNCA.git@84a71ee58ece03e3aaf5337b0c48d47a4f24a65f |
| AUTOVALIDATE_R_SOURCE_SUBDIR | . |
| AUTOVALIDATE_R_STRICT | true |
| AUTOVALIDATE_R_SUBMIT | false |
| AUTOVALIDATE_R_UTILS_BRANCH | master |
| AUTOVALIDATE_R_UTILS_REPO | autovalidate-r/autovalidateutils |
| CI | true |
| NOT_CRAN | true |
| PORTAL_DRYRUN | |
| PORTAL_GITLAB_PIPELINE_FILE | .gitlab-ci.yml |
| PORTAL_GITLAB_PIPELINE_REF | latest |
| PORTAL_GITLAB_PIPELINE_REPO | autovalidate-r/gitlab-ci |
| PORTAL_METADATA_VARS_FILTER_REGEXP | ^(AUTOVALIDATE\|_R\|PORTAL) |
| PORTAL_REF_R_ENVIRONMENT_IMAGE | registry.code.roche.com/autovalidate-r/autovalidate-r_images:R4.4.3 |
| PORTAL_REMEDIATION_ACTIVE | true |

## 6.2 Tools traceability data

| Tool | Value |
|---|---|
| Submission Portal commit SHA | 9392175593102 57b01606c490a7a5302371b14e4 |
| Pipeline commit SHA | b49fa102f841b9d00ad741eec1fefa1846a28bce |
| Autovalidate image digest | sha256:6ddab1da10b1fce60e9b6167c011e4dcb030d2f8735ea23f03acc629382d5550 |

| Tool | Value |
| --- | --- |
| Validation image digest | sha256:f196fe9b55e90ed628c2d2bbcf7b3f2a31188dc0095448291b4006fbd983f784 |
| Report template commit SHA | 95ed299814893cf78e38a6fc8bf79c87abb23517 |