

Package: aNCA (via r-universe)

October 13, 2026

Title (Pre-)Clinical NCA in a Dynamic Shiny App

Version 0.1.0.9208

Description An interactive 'shiny' application for performing non-compartmental analysis (NCA) on pre-clinical and clinical pharmacokinetic data. The package builds on 'PKNCA' for core estimators and provides interactive visualizations, CDISC outputs ('ADNCA', 'PP', 'ADPP'), configurable TLGs (tables, listings, and graphs) and draft slide decks. Typical use cases include exploratory analysis, validation, reporting or teaching/demonstration of NCA methods. Methods and core estimators are described in Denney, Duvvuri, and Buckeridge (2015) ``Simple, Automatic Noncompartmental Analysis: The PKNCA R Package" <doi:10.1007/s10928-015-9432-2>.

License Apache License (>= 2)

Imports bslib, dplyr, formatters, ggplot2, glue, grid, htmltools, htmlwidgets, logger, magrittr, PKNCA (>= 0.12.1), plotly (>= 4.11.0), purrr, reactable, reactable.extras, rlang, shiny, shinycssloaders, shinyjs, shinyjqui, shinyWidgets, stats, tidyr, tools, units, writexl, yaml, zip

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Suggests arrow, covr, DT, flextable, ggh4x, haven, jsonlite, knitr, lintr (>= 3.2.0), markdown, mockery, officer, pak, readxl, rlistings, rmarkdown, sass, scales, shinytest2, testthat (>= 3.0.0), usethis, quarto, vdiff, withr

Config/testthat/edition 3

Language en-US

URL <https://pharmaverse.github.io/aNCA/>,
<https://github.com/pharmaverse/aNCA>

BugReports <https://github.com/pharmaverse/aNCA/issues>

Depends R (>= 4.4), utils

LazyData true

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev libudunits2-dev zlib1g-dev

Repository <https://pharmaverse.r-universe.dev>

Date/Publication 2026-10-13 15:48:33 UTC

RemoteUrl <https://github.com/pharmaverse/anca>

RemoteRef HEAD

RemoteSha f8d921bfc8abc0b8ab0cbac6fe65cba58efdefed

Contents

.compress_range	5
.create_filter_expr	5
.eval_range	6
.handle_tooltips	6
add_exclusion_reasons	7
add_label_attribute	7
add_pptx_sl_plot	8
add_pptx_sl_plottable	9
add_pptx_sl_table	9
add_qmd_plot	10
add_qmd_sl_plot	10
add_qmd_sl_plottabletable	11
add_qmd_table	11
add_settings_version	12
adjust_class_and_length	12
adnca_example	13
apply_filters	14
apply_labels	15
apply_mapping	16
apply_unit_defaults	17
calculate_ratio_app	18
calculate_ratios	19
calculate_summary_stats	20
calculate_table_ratios	21
cdisc_validation_blocks_save	21
changed_units	22
convert_volume_units	22
create_metabfl	24
create_pptx_dose_slides	24
create_qmd_doc	25
create_qmd_dose_slides	26
create_settings_version	26
create_start_impute	27
delete_settings_version	27

detect_study_types	28
dm_example	30
dose_profile_duplicates	31
ex_example	32
exploration_individualplot	33
exploration_meanplot	34
export_cdisc	36
export_validation_blocks_save	37
filter_breaks	37
flexible_violinboxplot	38
format_pkncaconc_data	39
format_pkncadata_intervals	41
format_pkncadose_data	42
format_unit_string	43
g_lineplot	43
g_pkcg01_lin	45
g_pkcg01_log	46
g_pkcg01_sbs	46
g_pkcg02_lin	47
g_pkcg02_log	47
g_pkcg02_sbs	48
g_pkcg03_lin	48
g_pkcg03_log	49
g_pkcg03_sbs	49
generate_pre_specs	50
generate_tooltip_text	50
get_conversion_factor	51
get_half-life_plots	52
get_label	52
get_settings_code	53
interval_add_impute	54
interval_remove_impute	55
keep_blq_timepoints	56
l_pkcl01	57
l_pkcl01_tad	59
l_pkcl02_uri	60
l_pkpl01	61
l_pkpl04_mp	62
metadata_nca_parameters	63
metadata_nca_variables	64
multiple_matrix_ratios	65
p_pkcg03_lin_dose	66
p_pkcg03_log_dose	67
p_pkcg03_sbs_dose	67
p_pkpg01_cum	68
p_pkpg02_doseprop	69
p_pkpg03_boxp	71
p_pkpg06_mp	72

parse_annotation	73
pc_example	73
pivot_wider_pknca_results	74
pk.calc.ermx	75
pk.calc.ertlst	76
pk.calc.ertmax	76
pk.calc.volpk	77
pk_dose_qc_plot	78
pkcg01	80
pkcg02	82
pkcg03	84
PKNCA_build_units_table	86
PKNCA_calculate_nca	87
PKNCA_create_data_object	88
PKNCA_hl_rules_exclusion	90
PKNCA_impute_method_start_c1	91
PKNCA_impute_method_start_logslope	92
PKNCA_update_data_object	93
prepare_plot_data	95
read_pk	96
read_settings	97
read_versioned_settings	97
remove_pp_not_requested	98
rm_impute_obs_params	98
run_app	99
settings_version_summary	100
simplify_unit	101
t_pkct01	101
t_pkpt03_col	103
t_pkpt07_norm	105
t_pkpt08_uri	106
t_pkpt11_gmr	108
translate_terms	109
update_main_intervals	110
update_pknca_with_rules	111
validate_cdisc_types	111
validate_export_outputs	112
validate_pk	113
write_versioned_settings	114

.compress_range	<i>Compresses a numeric vector into the simplest possible character string that, when evaluated, will create the same numeric vector.</i>
-----------------	---

Description

Compresses a numeric vector into the simplest possible character string that, when evaluated, will create the same numeric vector.

Usage

```
.compress_range(range_vector)
```

Arguments

range_vector numeric vector with numbers to compress into string

Value

simplest possible character string representing provided vector

.create_filter_expr	<i>Helper function create text used to filter data frame</i>
---------------------	--

Description

Helper function create text used to filter data frame

Usage

```
.create_filter_expr(boxplotdata, varvalstofilter)
```

Arguments

boxplotdata Data frame to be filtered
varvalstofilter Character vector specifying which variable and value to pre-filter as colname: value. By default is NULL (no pre-filtering)

Value

The filter expression

<code>.eval_range</code>	<i>Evaluates range notation. If provided notation is invalid, returns NA.</i>
--------------------------	---

Description

Evaluates range notation. If provided notation is invalid, returns NA.

Usage

```
.eval_range(x)
```

Arguments

<code>x</code>	character string with range notation, e.g. 1:5.
----------------	---

Value

numeric vector with specified range of numbers, NA if notation is invalid

<code>.handle_tooltips</code>	<i>Tooltip construction helper</i>
-------------------------------	------------------------------------

Description

Tooltip construction helper

Usage

```
.handle_tooltips(data, tooltip_vars, labels_df)
```

Arguments

<code>data</code>	the dataframe
<code>tooltip_vars</code>	character vector of tooltip variables to extract
<code>labels_df</code>	data.frame used for label lookups in tooltips

Value

data with added tooltip_text column

add_exclusion_reasons *Add Exclusion Reasons to PKNCAdata Object*

Description

This function adds exclusion reasons to the exclude column of the concentration object within a PKNCAdata object, based on a list of reasons and row indices.

Usage

```
add_exclusion_reasons(pknca_data, exclusion_list)
```

Arguments

pknca_data A PKNCAdata object.

exclusion_list A list of lists, each with elements:

- reason: character string with the exclusion reason
- rows: integer vector of row indices to apply the reason to
- exclude_nca: logical, if TRUE the rows are excluded from NCA calculations (added to the exclude column)
- exclude_tlg: logical, if TRUE the rows are flagged with PKSUMXF = "Y" and the reason is stored in PKSUM1RS so TLGs can filter them out

Value

The modified PKNCAdata object with updated exclusion reasons, PKSUMXF, and PKSUM1RS in the concentration object.

add_label_attribute *Helper function to add "label" attribute to columns based on parameter names.*

Description

Helper function to add "label" attribute to columns based on parameter names.

Usage

```
add_label_attribute(df, myres)
```

Arguments

df Data frame to which labels will be added.

myres The output of PKNCA::pk.nca.

Value

Data frame with "label" attributes added to specified columns.

Examples

```
# 1. Setup example data
df <- data.frame(a = c(5.2, 6.3), b = c(50.1, 60.2))
names(df) <- c("CMAX[ng/mL]", "AUCLST[hr*ng/mL]")

# 2. Setup mock metadata (myres)
myres <- list(
  result = data.frame(
    Pptestcd = c("CMAX", "AUCLST"),
    PPSTRESU = c("ng/mL", "hr*ng/mL"),
    type_interval = "main",
    start = 0,
    end = 24
  )
)

# 3. Apply labels
labeled_df <- add_label_attribute(df, myres)

# 4. Check label
attr(labeled_df[["CMAX[ng/mL]"]], "label")
```

add_pptx_sl_plot	<i>Add a slide with a plot only</i>
------------------	-------------------------------------

Description

Add a slide with a plot only

Usage

```
add_pptx_sl_plot(pptx, plot)
```

Arguments

pptx	rpptx object
plot	ggplot object to show as plot

Value

rpptx object with slide added

add_pptx_sl_plottable *Add a slide with both a plot and a table*

Description

Add a slide with both a plot and a table

Usage

```
add_pptx_sl_plottable(pptx, df, plot)
```

Arguments

pptx	rpptx object
df	Data frame to show as table
plot	ggplot object to show as plot

Value

rpptx object with slide added

add_pptx_sl_table *Add a slide with a table only*

Description

Add a slide with a table only

Usage

```
add_pptx_sl_table(  
  pptx,  
  df,  
  title = "",  
  subtitle = "",  
  footer = "Click here for individual results"  
)
```

Arguments

pptx	rpptx object
df	Data frame to show as table
title	Title text for the slide
subtitle	Subtitle text for the slide
footer	Footer text for the slide

Value

rpptx object with slide added

add_qmd_plot	<i>Helper to create a Quarto code chunk for a plot</i>
--------------	--

Description

Used internally to format a plot chunk for Quarto documents.

Usage

```
add_qmd_plot(plot_expr, use_plotly = FALSE)
```

Arguments

- plot_expr Expression for plot.
- use_plotly Logical, whether to convert plot to plotly.

Value

Character vector for Quarto code chunk.

add_qmd_sl_plot	<i>Add a slide to the Quarto document with a single plot</i>
-----------------	--

Description

Used internally for dose escalation reporting.

Usage

```
add_qmd_sl_plot(quarto_path, plot, use_plotly = FALSE)
```

Arguments

- quarto_path Path to the Quarto (.qmd) file to append to.
- plot Expression for plot.
- use_plotly Logical, whether to convert plot to plotly.

Value

Invisibly returns TRUE if the slide was added.

`add_qmd_sl_plottabletable`*Add a slide to the Quarto document with a plot and two tables (side by side)*

Description

Used internally for dose escalation reporting.

Usage

```
add_qmd_sl_plottabletable(quarto_path, df1, df2, plot, use_plotly = FALSE)
```

Arguments

<code>quarto_path</code>	Path to the Quarto (.qmd) file to append to.
<code>df1</code>	Expression for first table (left column).
<code>df2</code>	Expression for second table (right column).
<code>plot</code>	Expression for plot.
<code>use_plotly</code>	Logical, whether to convert plot to plotly.

Value

Invisibly returns TRUE if the slide was added.

`add_qmd_table`*Helper to create a Quarto code chunk for a table*

Description

Used internally to format a table chunk for Quarto documents.

Usage

```
add_qmd_table(table_expr)
```

Arguments

<code>table_expr</code>	Expression for table.
-------------------------	-----------------------

Value

Character vector for Quarto code chunk.

`add_settings_version` *Add a new version to an existing versioned settings list*

Description

Prepends a new version entry so it becomes the current version. The previous current entry moves to the previous list.

Usage

```
add_settings_version(versions, new_version)
```

Arguments

<code>versions</code>	Existing list of version entries.
<code>new_version</code>	A single version entry (from <code>create_settings_version()</code>).

Value

Updated list of version entries with the new entry first.

`adjust_class_and_length`

Helper: adjust class and length (optional) for a data.frame based on metadata_nca_variables

Description

Helper: adjust class and length (optional) for a data.frame based on metadata_nca_variables

Usage

```
adjust_class_and_length(df, metadata, adjust_length = TRUE)
```

Arguments

<code>df</code>	Data frame to adjust.
<code>metadata</code>	Metadata data frame with variable specifications.
<code>adjust_length</code>	Logical indicating whether to adjust length of variables.

Details

Adjusts the class and length of each variable in the provided data frame according to the specifications in the metadata data frame. Only adjusts the length if `adjust_length = TRUE`

Value

Adjusted data frame.

adnca_example

adnca_example

Description

A pharmacokinetic concentration-time dataset containing multiple dose administration data with specimen types (SERUM and URINE), dosing information, and subject demographics. This is an example dataset for demonstrating pharmacokinetic analysis workflows. It follows the ADaMIG-ADNCA v1.0 structure.

Usage

```
adnca_example
```

Format

A data frame with 76 rows and 32 variables:

STUDYID Character. Study identifier.

USUBJID Character. Unique subject identifier.

PCSPEC Character. Specimen type (SERUM or URINE).

PARAM Character. Parameter name (e.g., DrugA).

METABFL Character. Metabolite flag indicator.

AFRLT Numeric. Actual relative time to first dose (hours).

NFRLT Numeric. Nominal relative time to first dose (hours).

ARRLT Numeric. Actual relative time to reference dose (hours).

NRRLT Numeric. Nominal relative time to reference dose (hours).

TRTRINT Numeric. Treatment interval in hours.

RRLTU Character. Relative reference time units.

AVAL Numeric. Actual analysis value (concentration).

AVALU Character. Analysis value units (ug/mL or mg/mL).

VOLUME Numeric. Volume of specimen collected (mL for urine).

VOLUMEU Character. Volume units.

DOSETRT Character. Dose treatment identifier.

TRT01A Character. Actual treatment received (formatted dose and route).

DOSEA Numeric. Actual dose amount (mg).

DOSEU Character. Dose units.

ROUTE Character. Route of administration.

ADOSEDUR Numeric. Actual dose duration (hours).

ATPTREF Character. Analysis time point reference (e.g., DOSE 1).

AGE Numeric. Subject age.

AGEU Character. Age units.

RACE Character. Race/ethnicity of subject.

SEX Character. Sex of subject.

NCA1XRS Character. NCA result code 1 (if applicable).

NCA2XRS Character. NCA result code 2 (if applicable).

NEFRLT Numeric. Nominal relative time to first dose (alternative column).

NERRLT Numeric. Nominal relative time to reference dose (alternative column).

AEFRLT Numeric. Actual relative time to first dose (alternative column).

AERRLT Numeric. Actual relative time to reference dose (alternative column).

DOSFRM Character. Dose formulation (e.g., "TABLET").

Details

This dataset contains pharmacokinetic data from multiple subjects receiving multiple dose regimens with different levels and routes of administration at 24-hour intervals. Concentration measurements are collected at multiple time points post-dose for both serum and urine samples. Analytes include parent drug and a metabolite.

Source

Example data created for package demonstration and testing purposes.

apply_filters	<i>Apply Filters to a Dataset</i>
---------------	-----------------------------------

Description

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

Usage

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

Arguments

data	A data frame containing the raw data to be filtered.
filters	A list of filters, where each filter is a list containing the column, condition, and value.

Details

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

Value

A data frame containing the filtered data.

Examples

```
# Example usage:
data <- mtcars
filters <- list(
  list(column = "mpg", condition = ">", value = "20"),
  list(column = "cyl", condition = "==", value = "6")
)
filtered_data <- apply_filters(data, filters)
```

 apply_labels

Apply Labels to a dataset

Description

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

Usage

```
apply_labels(data, labels_df = metadata_nca_variables, type = "ADNCA")
```

Arguments

data	The dataset to which labels will be applied.
labels_df	A data frame containing at least the columns "Variable", "Label", and "Dataset".
type	The type variable in labels_df for which the labels are to be applied.

Value

The same dataset with label attributes applied to all columns. If a column is not present in the labels list, it will be assigned the name of the col. If label already exists in the original data, it will be preserved.

Examples

```
data <- data.frame(USUBJID = c(1, 2, 3), AVAL = c(4, 5, 6))
labels <- data.frame(
  Variable = c("USUBJID", "AVAL"),
  Label = c("Unique Subject Identifier", "Analysis Value"),
  Dataset = c("ADNCA", "ADNCA")
)
data <- apply_labels(data, labels, "ADNCA")
print(attr(data$USUBJID, "label")) # "Unique Subject Identifier"
print(attr(data$AVAL, "label"))   # "Analysis Value"
```

 apply_mapping

Apply UI-Based Column Mapping to a Dataset

Description

This function takes a dataset and applies user-specified column mappings provided through a Shiny input object. It renames columns based on the selected mappings, handles special cases such as ADOSEDUR, updates units for key variables, applies labels, and removes detected duplicate concentration records.

Usage

```
apply_mapping(
  dataset,
  mapping,
  desired_order = MAPPING_DESIRED_ORDER,
  silent = TRUE,
  req_mappings = c("USUBJID", "AFRLT", "NFRLT", "ARRLT", "NRRLT", "PCSPEC", "ROUTE",
    "AVAL", "STUDYID", "ATPTREF", "AVALU", "RRLTU", "DOSEU", "PARAM")
)
```

Arguments

dataset	A data frame containing the raw data to be transformed.
mapping	A named list of column mappings.
desired_order	A character vector specifying the desired column order in the output dataset. Defaults to MAPPING_DESIRED_ORDER.
silent	Boolean, whether to print message with applied mapping. Defaults to TRUE.
req_mappings	A character vector indicating the names of the mapping object that must always be populated

Details

- Validates that all required columns are mapped and no duplicates exist.
- If ADOSEDUR is not mapped, it is assigned a value of 0.
- Removes concentration data duplicates using all columns except ARRLT, NRRLT, and ATPTREF.

Value

A transformed data frame with:

- Renamed columns according to user mappings
- Created columns for mapped non-existent variables (e.g., mg/L in AVALU)
- Created columns for unmapped required variables (e.g., ADOSEDUR)
- Ordered columns as specified
- Labels applied to columns (via `apply_labels()`)
- Concentration duplicates removed based on key identifiers: AFRLT, STUDYID, PCSPEC, DOSETRT, USUBJID, and PARAM

<code>apply_unit_defaults</code>	<i>Apply default target units onto a data-derived units table</i>
----------------------------------	---

Description

Takes a table of desired target units (with Pptestcd and PPstresu) and merges it with a data-derived units table to fill in PPORRESU and calculate conversion_factor. Parameters not present in the defaults retain their data-derived values.

Usage

```
apply_unit_defaults(default_units, data_units)
```

Arguments

<code>default_units</code>	A data frame with at least Pptestcd and PPstresu columns.
<code>data_units</code>	A data frame with the full data-derived units table (Pptestcd, PPORRESU, PPstresu, conversion_factor, and optional group columns).

Value

A list with two elements:

- `units` - The complete units table with defaults applied.
- `failed` - A data frame of rows where conversion factor could not be calculated.

calculate_ratio_app	<i>Links the table ratio of the App with the ratio calculations via PKNCA results</i>
---------------------	---

Description

Links the table ratio of the App with the ratio calculations via PKNCA results

Usage

```
calculate_ratio_app(
  res,
  test_parameter,
  ref_parameter = test_parameter,
  test_group = "(all other levels)",
  ref_group = "PARAM: Analyte01",
  aggregate_subject = "no",
  adjusting_factor = 1,
  custom_pptestcd = NULL
)
```

Arguments

res	A PKNCAresult object.
test_parameter	Character. The PPTESTCD value to use as test (numerator). May include a range suffix for interval parameters (e.g. AUCINT_0-20).
ref_parameter	Character. The PPTESTCD value to use as reference (denominator). Defaults to test_parameter. May include a range suffix.
test_group	Character. The test group (numerator). Default is "(all other levels)".
ref_group	Character. The reference group (denominator).
aggregate_subject	Character. Aggregation mode: "yes", "no", or "if-needed".
adjusting_factor	Numeric that multiplies the calculated ratio. Default is 1.
custom_pptestcd	Optional character. If provided, will be used as the PPTESTCD value.

Value

A data.frame with the calculated ratios for the specified settings.

calculate_ratios	<i>Calculate Ratios from PKNCA Results</i>
------------------	--

Description

Calculate Ratios from PKNCA Results

Usage

```
calculate_ratios(
  data,
  test_parameter,
  ref_parameter = test_parameter,
  match_cols,
  ref_groups,
  test_groups = NULL,
  adjusting_factor = 1,
  custom_pptestcd = NULL
)
```

Arguments

data	A PKNCAresults object or its result data.frame.
test_parameter	Character. The PPTESTCD value to use as test (numerator).
ref_parameter	Character. The PPTESTCD value to use as reference (denominator). Defaults to test_parameter.
match_cols	Character vector of column names to match between test and reference groups or a data.frame specifying columns and values.
ref_groups	A data.frame specifying reference groups. At its minimum, contains the contrast variable value(s) for the reference.
test_groups	A data.frame specifying test groups. Optional. By default is NULL, allowing rows not in ref_groups be used as test.
adjusting_factor	Numeric to multiply the ratio. Default is 1.
custom_pptestcd	Optional character. If provided, will be used as the PPTESTCD value.

Value

A data.frame result object with the calculated ratios.

`calculate_summary_stats`*Calculate Summary Statistics*

Description

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

Usage

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

Arguments

<code>data</code>	A data frame containing results of Non Compartmental Analysis using PKNCA package. Assumes presence of columns: PPORRES, PPSTRES, PPSTRESU, PPTESTCD
<code>input_groups</code>	A character vector specifying the columns to group by. Here. the hierarchical order matters Default is "PPSTRESU".

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.

Value

A data frame with summary statistics for each group and parameter.

Examples

```
data <- data.frame(
  ATPTREF = c(1, 1, 1, 1, 1, 1),
  PPTESTCD = c("A", "A", "B", "B", "C", "C"),
  PPORRES = c(10, 20, 5, 15, NA, 30),
  PPSTRES = c(10, 20, 5, 15, NA, 30),
  PPORRESU = c("mg/L", "mg/L", "ng/mL", "ng/mL", "µg/L", "µg/L"),
  PPSTRESU = c("mg/L", "mg/L", "ng/mL", "ng/mL", "µg/L", "µg/L")
)
calculate_summary_stats(data)
```

calculate_table_ratios

Apply Ratio Calculations to PKNCAResult Object

Description

This function takes a PKNCAResult object and a data.frame specifying ratio calculations.

Usage

```
calculate_table_ratios(res, ratio_table)
```

Arguments

res	A PKNCAResult object.
ratio_table	Data.frame with columns: TestParameter, RefParameter, Reference, Test, AggregateSubject, AdjustingFactor, TestGroups, RefGroups, PPTESTCD.

Value

The updated PKNCAResult object with added rows in the result data.frame.

cdisc_validation_blocks_save

Does a set of validation findings block saving?

Description

Predicate used by the export/Save flow to refuse a save when any error-severity finding is present. warning-severity findings (for example selected custom grouping columns) do not block saving.

Usage

```
cdisc_validation_blocks_save(findings)
```

Arguments

findings A findings data frame as returned by `validate_cdisc_types()`.

Value

TRUE when at least one error-severity finding exists, otherwise FALSE.

Examples

```
findings <- validate_cdisc_types(list())
cdisc_validation_blocks_save(findings)
```

changed_units	<i>Keep only units that differ from their data-derived default</i>
---------------	--

Description

Change detection is value-based: a row is considered "changed" when its target unit (PPSTRESU) differs from the original unit (PPORRESU). This replaces the previous `default` flag, which only tracked edits made through the Units modal and missed automatic changes such as volume simplification.

Usage

```
changed_units(units)
```

Arguments

units A units table with PPORRESU and PPSTRESU columns.

Value

The subset of rows where PPSTRESU differs from PPORRESU. Rows where either unit is NA are excluded to avoid emitting NA-based entries downstream.

convert_volume_units	<i>Convert Volume Units to Match Concentration Denominator Units</i>
----------------------	--

Description

This function identifies rows associated with excretion samples (e.g., urine, feces, bile) and adjusts the `VOLUME` and `VOLUMEU` columns so that the volume unit matches the denominator unit in the corresponding concentration unit (`AVALU`). This is necessary for PKNCA calculation of excretion parameters.

Usage

```
convert_volume_units(
  df,
  avalu = "AVALU",
  volume = "VOLUME",
  volumeu = "VOLUMEU"
)
```

Arguments

df	A data frame containing pharmacokinetic data.
avalu	A character string specifying the column name for concentration values (default: "AVALU").
volume	A character string specifying the column name for volume or mass values (default: "VOLUME").
volumeu	A character string specifying the column name for volume or mass units (default: "VOLUMEU"). It must contain the following columns: PCSPEC Sample type (e.g., urine, feces, bile, plasma). AVAL Concentration values. AVALU Concentration units (e.g., "ug/mL", "mg/g"). VOLUME Volume or mass values for integration. VOLUMEU Units for the VOLUME column (e.g., "mL", "g").

Details

It uses the **units** package to perform unit-safe conversions. If a direct conversion between volume and the concentration denominator is not possible (e.g., between mass and volume), a fallback conversion is attempted using a neutral density of 1 ($\text{target_unit} / \text{original_unit}$). The function modifies only the VOLUME and VOLUMEU columns when necessary and leaves all other data unchanged.

The function:

1. Parses the denominator from AVALU (e.g., "ug/mL" $\rightarrow$ "mL").
2. Attempts to convert the corresponding VOLUME to that unit.
3. If direct conversion fails, assumes a neutral density of 1 (i.e., $1 \text{ unit_target} / \text{unit_original}$) and retries.
4. Leaves units unchanged for non-excreta samples or already-valid combinations.

The function assumes that the AVALU column contains concentration units in the form of "x/y" (e.g., "ug/mL", "mg/g").

Value

A modified data frame with VOLUME and VOLUMEU converted (where necessary) so that multiplying $\text{AVAL} * \text{VOLUME}$ results in a unit with consistent dimensionality (typically mass or moles). A new column AMOUNTU is created to represent the product of AVALU and VOLUMEU.

Examples

```
df <- data.frame(
  PCSPEC = c("urine", "feces", "plasma"),
  AVAL = c(100, 5, 70),
  AVALU = c("ug/mL", "mg/g", "ng/mL"),
  VOLUME = c(2, 1.5, 3),
  VOLUMEU = c("L", "mL", "mL"),
  stringsAsFactors = FALSE
)

df_converted <- convert_volume_units(df)
```

create_metabfl	<i>Create METABFL Column Based on Selected Metabolites</i>
----------------	--

Description

Create METABFL Column Based on Selected Metabolites

Usage

```
create_metabfl(dataset, metabolites)
```

Arguments

dataset	A data frame containing a dataset with a PARAM (parameter) column.
metabolites	A character vector of parameter names representing metabolites.

Value

The input dataset with an additional METABFL column indicating metabolite records ("Y") or non-metabolite records ("").

create_pptx_dose_slides	<i>Create a PowerPoint presentation with dose escalation results, including main and extra figures Adds slides for summary tables, mean plots, line plots, and individual subject results</i>
-------------------------	---

Description

Create a PowerPoint presentation with dose escalation results, including main and extra figures
Adds slides for summary tables, mean plots, line plots, and individual subject results

Usage

```
create_pptx_dose_slides(res_dose_slides, path, title, template)
```

Arguments

res_dose_slides	List of results for each dose group
path	File path to save the presentation
title	Title for the presentation
template	Path to PowerPoint template file

Value

TRUE (invisible). Writes the PowerPoint file to the specified path

create_qmd_doc	<i>Create a new Quarto presentation file with YAML header and setup chunk</i>
----------------	---

Description

Used internally to initialize a Quarto document for exporting plots and tables.

Usage

```
create_qmd_doc(
  quarto_path,
  title = "NCA Report",
  libraries = c("plotly", "flextable", "dplyr"),
  rda_path = NULL,
  template = NULL,
  extra_setup = NULL
)
```

Arguments

quarto_path	Path to the Quarto (.qmd) file to create.
title	Title for the presentation.
libraries	Character vector of libraries to load in setup chunk.
rda_path	Path to the RDS file to be loaded in the document.
template	(Optional) Path to a Quarto template to use (default: NULL).
extra_setup	(Optional) Character vector of extra setup lines to include after YAML.

Value

Invisibly returns TRUE if the file was created.

```
create_qmd_dose_slides
```

Create all slides for dose escalation results in a Quarto document

Description

Used internally to generate main and individual slides for each dose group.

Usage

```
create_qmd_dose_slides(res_dose_slides, quarto_path, title, use_plotly = TRUE)
```

Arguments

<code>res_dose_slides</code>	List of results for each dose group.
<code>quarto_path</code>	Path to the Quarto (.qmd) file to create.
<code>title</code>	Title for the presentation.
<code>use_plotly</code>	Logical, whether to convert plots to plotly.

Value

Invisibly returns TRUE if slides were created.

```
create_settings_version
```

Create a versioned settings entry

Description

Wraps a settings payload with metadata (timestamp, comment, dataset name, aNCA version, active tab) for storage in a versioned settings YAML file.

Usage

```
create_settings_version(settings_data, comment = "", dataset = "", tab = "")
```

Arguments

<code>settings_data</code>	List with the settings payload (mapping, parameters, NCA setup, filters, slope_rules, etc.).
<code>comment</code>	Optional character string with a user comment.
<code>dataset</code>	Optional character string identifying the dataset used.
<code>tab</code>	Optional character string with the active app tab.

Value

A named list representing one version entry.

create_start_impute	<i>Create C0 Impute Column</i>
---------------------	--------------------------------

Description

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

Usage

```
create_start_impute(pknca_data)
```

Arguments

pknca_data A PKNCAdata object containing concentration and dose data.

Value

A PKNCAdata object with updated intervals table including start imputation strategies. If the intervals are ambiguous and can refer to multiple concentration groups it will define them to choose the proper imputation for each.

Examples

```
adnca <- adnca_example
pknca_data <- PKNCA_create_data_object(adnca)
pknca_data <- create_start_impute(pknca_data)
```

delete_settings_version	<i>Delete a version from a versioned settings list</i>
-------------------------	--

Description

Removes the version at the given index. Cannot delete the last remaining version.

Usage

```
delete_settings_version(versions, index)
```

Arguments

versions	List of version entries.
index	Integer index of the version to remove.

Value

Updated list of version entries.

detect_study_types	<i>Detect study types</i>
--------------------	---------------------------

Description

This function detects the type of study based on the provided data.

Usage

```
detect_study_types(
  data,
  groups,
  metabfl_column,
  route_column,
  volume_column = "volume"
)
```

Arguments

data	The dataset containing the study types to be identified. Assumed to be the output from aNCA formatted concentration data. Must contain group columns, the specified route, analyte and drug columns, and ADOSEDUR.
groups	the grouping variables for the study type detection.
metabfl_column	A character string specifying the column name for the metabolite flag, which is coded as "Y" for metabolites
route_column	A character string specifying the column name for the route of administration.
volume_column	A character string specifying the column name for the volume of a sample. Extravascular samples must be written as extravascular. Can be set to volume if not applicable.

Details

The function identifies a possible five different types of studies based on grouping by STUDYID, DOSETRT, USUBJID, PCSPEC, and the route column. The study types are determined as follows:

- "Excretion Data": If the volume column for a group is not NA and has values > 0.
- "Single Extravascular Dose": If there is only one dose and TRTRINT is not available, and the route is extravascular.
- "Single IV Infusion Dose": If there is only one dose and TRTRINT is not present, and the route is not extravascular.
- "Single IV Bolus Dose": If there is only one dose and TRTRINT is not present, the route is not extravascular, and ADOSEDUR == 0.

- "Multiple Extravascular Doses": If there are multiple doses (or TRTRINT is available) and the route is extravascular.
- "Multiple IV Infusion Doses": If there are multiple doses (or TRTRINT is available) and the route is not extravascular.
- "Multiple IV Bolus Doses": If there are multiple doses or TRTRINT is not present, the route is not extravascular, and ADOSEDUR == 0.
- If none of these conditions are met, the type is marker as "Unknown".

Value

A data frame with the unique combinations of study types, including the grouping columns and the identified type.

Examples

```
sample_data <- data.frame(
  STUDYID = "STUDY001",
  DOSETRT = "Drug",
  ANALYTE = "DOSETRT",
  USUBJID = c(
    # 1. Single IV Dose subject
    "Subj-01", "Subj-01",
    # 2. Multiple Extravascular Doses subject (identified by TRTRINT)
    "Subj-02", "Subj-02",
    # 3. Excretion Data subject (identified by positive volume)
    "Subj-03", "Subj-03"
  ),
  PCSPEC = "PLASMA",
  DOSNOA = c(
    1, 1,      # Single dose
    1, 1,      # Appears as single dose...
    1, 1       # Single dose
  ),
  ROUTE = c(
    "INTRAVENOUS", "INTRAVENOUS",
    "extravascular", "extravascular",
    "INTRAVENOUS", "INTRAVENOUS"
  ),
  ADOSEDUR = c(
    0, 0,
    0, 0,
    2, 2),
  SAMPLE_VOLUME = c(
    NA, 0,
    NA, 0,
    10, 12      # Positive volume indicates excretion
  ),
  TRTRINT = c(
    NA, NA,
    24, 24,     # ...but TRTRINT indicates a multiple-dose regimen
    NA, NA
```

```

),
METABFL = c(
  "N", "N",
  "N", "N",
  "Y", "Y"    # mark last subject as metabolite
)
)

study_summary <- detect_study_types(
  data = sample_data,
  groups = c("USUBJID", "PCSPEC", "DOSETRT"),
  metabfl_column = "METABFL",
  route_column = "ROUTE",
  volume_column = "SAMPLE_VOLUME"
)

```

dm_example

dm_example

Description

An SDTM DM-like demographics dataset derived from [adnca_example](#). Contains one row per subject with demographic variables and a synthetic treatment start date (RFXSTDTC) that serves as the reference origin for datetimes in [pc_example](#) and [ex_example](#).

Usage

```
dm_example
```

Format

A data frame with one row per subject and 9 variables:

DOMAIN Character. Domain Abbreviation.

STUDYID Character. Study identifier.

USUBJID Character. Unique subject identifier.

AGE Numeric. Age of the subject.

AGEU Character. Age units (e.g., "Years").

SEX Character. Sex of the subject.

RACE Character. Race of the subject.

ARM Character. Planned treatment arm (mapped from TRT01A).

ACTARM Character. Actual treatment arm (mapped from TRT01A).

RFXSTDTC Character. Date/time of first study treatment (ISO 8601). Synthetic, staggered by 3 days per subject to simulate different enrollment dates.

Source

Derived from [adnca_example](#) via `data-raw/sdtm_example.R`.

dose_profile_duplicates

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

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.

Usage

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

Arguments

<code>conc_data</code>	A data frame containing concentration data.
<code>groups</code>	A character vector of column names to use for grouping (e.g., <code>c("USUBJID", "PARAM", "DOSNOA")</code>).
<code>dosno</code>	Column name for the dose number (default: "DOSNOA").
<code>arrlt</code>	Column name for time from the most recent dose.
<code>afrlt</code>	Column name for time from the first dose.
<code>nrslt</code>	Column name for the numeric relative time.
<code>nfrlt</code>	Column name for the nominal relative time.

Value

A data frame with adjusted concentration data, including:

- Duplicated pre-dose values assigned to the previous dose.
- Duplicated last values assigned to the next dose if pre-dose values are missing, filtered so only samples within 4 hours of the next dose are included.
- Sorted by the grouping variables and relative time.

Examples

```
# Example usage
conc_data <- data.frame(
  USUBJID = c("001", "001", "001", "001", "001", "001", "001", "001", "001", "001"),
  AVAL = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
  DOSNOA = c(1, 1, 1, 2, 2, 2, 2, 3, 3, 3),
  ARRLT = c(-1, 0, 1, -1, 0, 1, 2, 0, 1, 2),
  AFRLT = c(-1, 0, 1, 2, 3, 4, 5, 6, 7, 8),
  NRRLT = c(-1, 0, 1, -1, 0, 1, 2, 0, 1, 2),
  NFRLT = c(-1, 0, 1, 2, 3, 4, 5, 6, 7, 8)
)
result <- dose_profile_duplicates(conc_data,
  groups = c("USUBJID", "DOSNOA"), dosno = "DOSNOA")
```

ex_example

ex_example

Description

An SDTM EX-like exposure/dosing dataset derived from [adnca_example](#). Contains one row per dosing event per subject. Intended to support and test the PC+EX upload workflow (see GitHub issue #624).

Usage

```
ex_example
```

Format

A data frame with 11 variables:

DOMAIN Character. Domain abbreviation.

STUDYID Character. Study identifier.

USUBJID Character. Unique subject identifier.

EXTRT Character. Name of the treatment (mapped from DOSETRT).

EXDOSE Numeric. Dose amount (mapped from DOSEA).

EXDOSU Character. Dose units (mapped from DOSEU).

EXROUTE Character. Route of administration (mapped from ROUTE).

EXSTDTC Character. Dosing start datetime (ISO 8601), synthesised from AFRLT - ARRLT using the subject's RFXSTDTC as reference origin.

EXENDTC Character. Dosing end datetime (ISO 8601), computed as EXSTDTC + ADOSEDUR.

EXDUR Character. Duration of dose in ISO 8601 duration format (e.g., "PT2.9H"), derived from ADOSEDUR.

EXELTM Character. Planned elapsed time from first dose in ISO 8601 duration format (e.g., "PT0H"), derived from NFRLT - NRRLT relative to the subject's first dose.

Source

Derived from [adnca_example](#) and [dm_example](#) via data-raw/sdtm_example.R.

```
exploration_individualplot
```

Create an Individual PK Line Plot

Description

Generates a line plot for individual pharmacokinetic (PK) concentration-time profiles

Usage

```
exploration_individualplot(
  pknca_data,
  color_by,
  facet_by = NULL,
  show_facet_n = FALSE,
  ylog_scale = FALSE,
  lock_y_axis = FALSE,
  threshold_value = NULL,
  x_limits = NULL,
  y_limits = NULL,
  show_dose = FALSE,
  palette = "default",
  tooltip_vars = NULL,
  labels_df = NULL,
  filtering_list = NULL,
  use_time_since_last_dose = FALSE,
  show_legend = TRUE,
  line_type = "default"
)
```

Arguments

pknca_data	A PKNCAdata object containing the raw PK concentration data.
color_by	Character vector specifying the column(s) used to color the lines and points.
facet_by	Character vector of column names to facet the plot by. Default is NULL (no faceting).
show_facet_n	Logical; if TRUE, shows the number of subjects in each facet. Default is FALSE.
ylog_scale	Logical; whether to use a logarithmic scale for the y-axis. Default is FALSE.
lock_y_axis	Logical; if TRUE, faceted plots use one shared y-axis range while x-axis scales remain free. Default is FALSE.
threshold_value	Numeric; y-intercept for a horizontal threshold line. Default is NULL (no threshold).

x_limits	Numeric vector of length 2 for x-axis limits (min, max). Default is NULL (no limits).
y_limits	Numeric vector of length 2 for y-axis limits (min, max). Default is NULL (no limits).
show_dose	Logical; if TRUE, vertical lines for dose times are shown. Default is FALSE.
palette	Optional palette name or named character vector of colors for the plot. Default is "default" color palette.
tooltip_vars	Character vector of column names to include in the tooltip. Default is NULL.
labels_df	Optional data.frame for variable label lookups (for tooltips). Default is NULL.
filtering_list	Named list of filters (column = allowed values). Default is NULL (no filtering).
use_time_since_last_dose	Logical; if TRUE, x-axis represents time since last dose. Default is FALSE (time since first dose).
show_legend	Logical; whether to display the plot legend. Default is TRUE.
line_type	Character; "default" (default), "dose-normalized" to specify lines to be normalized by dose amount or "both" to include both normalized and non-normalized lines.

Value

A ggplot object representing the individual PK line plot.

exploration_meanplot *Create a Mean PK Line Plot*

Description

Generates a line plot for mean pharmacokinetic (PK) concentration-time profiles

Usage

```
exploration_meanplot(
  pknc_data,
  color_by,
  facet_by = NULL,
  show_facet_n = FALSE,
  ylog_scale = FALSE,
  lock_y_axis = FALSE,
  show_legend = TRUE,
  threshold_value = NULL,
  show_dose = FALSE,
  palette = "default",
  sd_min = FALSE,
  sd_max = FALSE,
```

```

    ci = FALSE,
    tooltip_vars = NULL,
    labels_df = NULL,
    filtering_list = NULL,
    use_time_since_last_dose = FALSE,
    x_limits = NULL,
    y_limits = NULL,
    line_type = "default"
  )

```

Arguments

<code>pknca_data</code>	A PKNCAdata object containing the raw PK concentration data.
<code>color_by</code>	Character vector specifying the column(s) used to color the lines and points.
<code>facet_by</code>	Character vector of column names to facet the plot by. Default is NULL (no faceting).
<code>show_facet_n</code>	Logical; if TRUE, shows the number of subjects in each facet. Default is FALSE.
<code>ylog_scale</code>	Logical; whether to use a logarithmic scale for the y-axis. Default is FALSE.
<code>lock_y_axis</code>	Logical; if TRUE, faceted plots use one shared y-axis range while x-axis scales remain free. Default is FALSE.
<code>show_legend</code>	Logical; whether to display the plot legend. Default is TRUE.
<code>threshold_value</code>	Numeric; y-intercept for a horizontal threshold line. Default is NULL (no threshold).
<code>show_dose</code>	Logical; if TRUE, vertical lines for dose times are shown. Default is FALSE.
<code>palette</code>	Optional palette name or named character vector of colors for the plot. Default is "default" color palette.
<code>sd_min</code>	Logical; if TRUE, plot lower SD error bars. Default is FALSE.
<code>sd_max</code>	Logical; if TRUE, plot upper SD error bars. Default is FALSE.
<code>ci</code>	Logical; if TRUE, plot 95% confidence interval ribbon. Default is FALSE.
<code>tooltip_vars</code>	Character vector of column names to include in the tooltip. Default includes dose group vars and "AVAL" (renamed from Mean).
<code>labels_df</code>	Optional data.frame for variable label lookups (for tooltips). Default is NULL.
<code>filtering_list</code>	Named list of filters (column = allowed values). Default is NULL (no filtering).
<code>use_time_since_last_dose</code>	Logical; if TRUE, x-axis represents time since last dose. Default is FALSE (time since first dose).
<code>x_limits</code>	Numeric vector of length 2 for x-axis limits (min, max). Default is NULL (no limits).
<code>y_limits</code>	Numeric vector of length 2 for y-axis limits (min, max). Default is NULL (no limits).
<code>line_type</code>	Character; "default" (default), "dose-normalized" to specify lines to be normalized by dose amount or "both" to include both normalized and non-normalized lines.

Value

A ggplot object representing the mean PK line plot, with error bars and/or confidence intervals if requested.

 export_cdisc

Export CDISC Data

Description

This function processes the results from a PKNCA and exports them into CDISC compliant datasets. Attention: All parameters that do not match pptest dataframe will be lost in this pipeline!

Usage

```
export_cdisc(res_nca, grouping_vars = character(0), flag_rules = NULL)
```

Arguments

res_nca	Object with results of the NCA analysis. If <code>res_nca\$result</code> contains a <code>.pp_excl</code> column (logical), excluded rows are merged into the <code>exclude</code> column so they appear in PPSUMXF/PPSUMRSN. If <code>.pp_excl_reason</code> (character) is also present, it populates PPSUMRSN.
grouping_vars	Character vector of non-standard grouping variable names to include as additional columns in ADNCA, ADPP, and PP outputs. Defaults to <code>character(0)</code> .
flag_rules	Character vector of flag rule exclusion messages applied during NCA (e.g., <code>c("R2ADJ < 0.8", "AUCPE0 > 20")</code>). Each entry generates a CRITy/CRITyFL column pair in ADPP, plus PPSUMXF and PPSUMRSN columns. Defaults to NULL (no flags).

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)

Value

A list with two data frames:

pp A data frame containing the PP (Pharmacokinetic Parameters) domain data.

adpp A data frame containing the ADPP (Analysis Dataset for Pharmacokinetic Parameters) domain data.

export_validation_blocks_save	<i>Do export findings block saving?</i>
-------------------------------	---

Description

Predicate used by the export/Save flow to refuse a save when any error-severity finding is present in the output of `validate_export_outputs()`.

Usage

```
export_validation_blocks_save(findings)
```

Arguments

`findings` A findings data frame as returned by `validate_export_outputs()`.

Value

TRUE when at least one error-severity finding exists, otherwise FALSE.

Examples

```
findings <- validate_export_outputs(list())
export_validation_blocks_save(findings)
```

filter_breaks	<i>Filter Breaks for X-Axis</i>
---------------	---------------------------------

Description

Filters an axis for consecutive breaks that are far enough apart to be drawn without their labels colliding. A break is kept when its distance from the last kept break is at least `min_cm_distance` *and* at least the space the two labels occupy along the axis, so a wide label such as 119.917 thins the axis further than a short one such as 12 does.

Usage

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

Arguments

breaks	A numeric vector of x-axis breaks.
plot	A ggplot object used to extract plot dimensions and scales.
min_cm_distance	A numeric of the minimum distance between breaks.
axis	Axis to filter on, either "x" or "y".
labels	A function turning breaks into the labels that will be drawn, used to measure how much room each one needs. Defaults to the same labels the concentration plots draw.

Details

Candidates outside the plotted range are dropped before any of that, so a break that is never drawn cannot decide which of the visible ones survive.

Value

A numeric vector of filtered x-axis breaks.

Author(s)

Gerardo Rodriguez

flexible_violinboxplot

Flexible Violin/Box Plot

Description

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

Usage

```
flexible_violinboxplot(
  res_nca,
  parameter,
  xvars,
  colorvars,
  varvalstofilter = NULL,
  tooltip_vars,
  labels_df = metadata_nca_variables,
  box = TRUE,
  plotly = TRUE,
  seed = NULL,
  show_excluded = FALSE
)
```

Arguments

res_nca	A PKNCA results object containing the results and concentration data.
parameter	A string specifying the parameter to be plotted.
xvars	Character vector of column names for the x-axis grouping.
colorvars	Character vector of column names for the color aesthetic.
varvalstofilter	Character vector specifying which variable and value to pre-filter as colname:value. By default is NULL (no pre-filtering)
tooltip_vars	A character vector indicating the column names from result_data that should be used to identify when hovering the plotly outputs.
labels_df	A data.frame used for label lookups in tooltips. Defaults to metadata_nca_variables.
box	A logical value indicating whether to plot a box plot (TRUE) or a violin plot (FALSE). Default is TRUE.
plotly	A logical value defining if the output is plotly (TRUE, default) or ggplot otherwise (FALSE)
seed	An integer value to set the seed for reproducibility of jittering. Default (NULL) will use the current R seed.
show_excluded	Logical. If TRUE, excluded records (those with a populated exclude column) are overlaid as cross-shaped points. They are never included in box/violin statistics. Default is FALSE.

Value

A plotly object representing the violin or box plot.

format_pkncaconc_data *Create PK Concentration Dataset*

Description

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

Usage

```
format_pkncaconc_data(
  ADNCA,
  group_columns,
  time_column = "AFRLT",
  rrlt_column = "ARRLT",
  route_column = "ROUTE",
  time_end_column = NULL,
  nca_exclude_reason_columns = NULL,
  dose_group_columns
)
```

Arguments

ADNCA	A data frame containing the ADNCA data.
group_columns	A character vector specifying the columns to group by.
time_column	A character string specifying the time column.
rrlt_column	A character string specifying the time since last dose column.
route_column	A character string specifying the route column.
time_end_column	(Optional) A character string specifying the column for end time, used to calculate CONCDUR.
nca_exclude_reason_columns	A character vector specifying the columns to indicate reasons for excluding records from NCA analysis.
dose_group_columns	Character vector of columns to group by when computing DOSNOA. Should use dose-level grouping (without specimen/analyte columns) to ensure consistent dose numbering across specimen types.

Details

The function performs the following steps:

- Checks for required columns and data.
- Filters out rows with EVID = 0 and PARAMCD containing "DOSE" (removes dosing data; not CDISC standard).
- Optionally creates CONCDUR if time_end_column is provided.
- Optionally creates nca_exclude by merging specified exclusion reason columns.
- Creates DOSNOA variable: sequential numbers based on time of dose within each group.
- Creates a std_route column with PKNCA values "intravascular" or "extravascular" based on the route column (ROUTE, CDISC: C66729).
- Arranges the data by group_columns and dose time.

Value

A data frame containing the filtered and processed concentration data. The following new columns are created:

- DOSNOA: Sequential dose number within each group, based on time of dose.
- std_route: Standardized route, either "intravascular" or "extravascular", derived from the route column.
- nca_exclude: Character column with concatenated exclusion reasons (if nca_exclude_reason_columns is provided), otherwise empty string.
- CONCDUR: (Optional) Sampling duration, calculated as time_end_column - time_column if time_end_column is provided.

Examples

```

adnca <- adnca_example
conc_data <- format_pkncaconc_data(
  ADNCA = adnca,
  group_columns = c("STUDYID", "DOSETRT", "USUBJID", "PARAM"),
  time_column = "AFRLT",
  rrlt_column = "ARRLT",
  route_column = "ROUTE",
  dose_group_columns = c("STUDYID", "DOSETRT", "USUBJID")
)

```

format_pkncadata_intervals

Create Dose Intervals Dataset

Description

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

Usage

```

format_pkncadata_intervals(
  pknc_conc,
  pknc_dose,
  start_from_last_dose = TRUE,
  keep_interval_cols = NULL
)

```

Arguments

pknc_conc	A PKNCAdose object containing the concentration data.
pknc_dose	A PKNCAdose object containing the dose data.
start_from_last_dose	Logical defining if start is at time of last dose or C1.
keep_interval_cols	Optional character vector of additional columns to keep in the intervals data frame.

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 TRTRINT column is present in data, if TRTRINT is NA then it sets end time to Inf if only one dose present, or max end time if not.

- If no TRTRINT column in data, sets last dose end time to the time of last sample or Inf if single dose data.
- Adds logical columns for each specified parameter.

Assumes that multiple dose data will have a TRTRINT column or contain multiple doses in dataset

Value

A data frame containing the dose intervals and specified pharmacokinetic parameters.

Examples

```
adnca <- adnca_example
pknca_data <- PKNCA_create_data_object(adnca)
pknca_conc <- pknca_data$conc
pknca_dose <- pknca_data$dose
dose_intervals <- format_pkncadata_intervals(pknca_conc, pknca_dose)
```

format_pkncadose_data *Create PK Dose Dataset*

Description

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

Usage

```
format_pkncadose_data(
  pkncaconc_data,
  time_column = "AFRLT",
  rrllt_column = "ARRLT",
  group_columns
)
```

Arguments

pkncaconc_data A data frame containing the concentration data.
time_column A character string specifying the time from first dose column.
rrllt_column A character string specifying the time since last dose column.
group_columns A character vector specifying the columns to group by.

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.

Value

A data frame containing the dose data.

format_unit_string	<i>Formats a unit string if a unique unit exists</i>
--------------------	--

Description

Formats a unit string if a unique unit exists

Usage

```
format_unit_string(data, unit_var)
```

Arguments

data	The data frame to check.
unit_var	The column name of the unit variable.

Value

A formatted string like "(hr)" or an empty string "".

g_lineplot	<i>Generate a Unified Line Plot for PK Data</i>
------------	---

Description

This function creates a ggplot2 line plot for pharmacokinetic (PK) data. The function supports various customizations including log scales, faceting and threshold lines.

Usage

```
g_lineplot(  
  data,  
  x_var,  
  y_var,  
  x_unit = NULL,  
  y_unit = NULL,  
  color_by,  
  color_labels = NULL,  
  facet_by = NULL,  
  group_by = NULL,  
  facet_count_n = NULL,  
  x_limits = NULL,
```

```

y_limits = NULL,
ylog_scale = FALSE,
lock_y_axis = FALSE,
threshold_value = NULL,
palette = "default",
tooltip_vars = NULL,
labels_df = NULL,
vline_var = NULL,
linetype_by = NULL,
show_legend = TRUE
)

```

Arguments

<code>data</code>	A data.frame containing the data to be plotted. This should be pre-processed by either <code>process_data_individual</code> or <code>process_data_mean</code> .
<code>x_var</code>	A character string specifying the column name for the x-axis.
<code>y_var</code>	A character string specifying the column name for the y-axis.
<code>x_unit</code>	Optional character string specifying the column name for the x-axis unit.
<code>y_unit</code>	Optional character string specifying the column name for the y-axis unit.
<code>color_by</code>	A character vector specifying the column(s) from the original dataset that are used to determine the color of the lines and points.
<code>color_labels</code>	Optional character vector of labels for the color legend. Default is NULL (uses <code>color_by</code> values).
<code>facet_by</code>	A character vector of column names to facet the plot by. Default is NULL for no faceting.
<code>group_by</code>	A character vector specifying the column names used to group the lines. Default is NULL for no grouping.
<code>facet_count_n</code>	A character string specifying the column name used to count unique subjects per facet. Default is NULL (no counts shown).
<code>x_limits</code>	Numeric vector of length 2 for x-axis limits (min, max). Default is NULL (no limits).
<code>y_limits</code>	Numeric vector of length 2 for y-axis limits (min, max). Default is NULL (no limits).
<code>ylog_scale</code>	A logical value (TRUE or FALSE) indicating whether to use a logarithmic scale for the y-axis.
<code>lock_y_axis</code>	Logical; if TRUE, faceted plots use one shared y-axis range while x-axis scales remain free. Default is FALSE.
<code>threshold_value</code>	A numeric value for the y-intercept of the threshold line. Only used if <code>show_threshold</code> is TRUE.
<code>palette</code>	A character string specifying the color palette to use. Default is "default" palette.
<code>tooltip_vars</code>	Character vector of column names to include in the tooltip.
<code>labels_df</code>	A data.frame for variable label lookups.

vline_var	Optional character string specifying the column name for vertical lines.
linetype_by	Optional character vector specifying the column name for line types.
show_legend	Logical; whether to display the plot legend. Default is TRUE.

Value

A ggplot object representing the line plot.

Examples

```
library(dplyr)
ind_data <- expand.grid(
  time_var = c(0, 1, 2, 4, 8, 12),
  USUBJID = c("Subject1", "Subject2")
) %>%
  mutate(
    AVAL = ifelse(USUBJID == "Subject1", 50, 80) * exp(-0.5 * time_var) + rnorm(n(), 0, 1),
    PARAM = "Analyte1",
    DOSEA = "Dose 1",
    RRLTU = "hours",
    AVALU = "ng/mL"
  )

p <- g_lineplot(
  data = ind_data,
  x_var = "time_var",
  y_var = "AVAL",
  color_by = "USUBJID"
)
print(p)
```

g_pkcg01_lin	<i>Wrapper around aNCA::pkcg01() function. Calls the function with LIN scale argument.</i>
--------------	--

Description

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

Usage

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

Arguments

data	Data to be passed into the plotting function.
...	Any other parameters to be passed into the plotting function.

Value

ggplot2 object for pkcg01.

g_pkcg01_log	<i>Wrapper around aNCA::pkcg01() function. Calls the function with LOG scale argument.</i>
--------------	--

Description

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

Usage

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

Arguments

data	Data to be passed into the plotting function.
...	Any other parameters to be passed into the plotting function.

Value

ggplot2 object for pkcg01.

g_pkcg01_sbs	<i>Wrapper around aNCA::pkcg01() function. Calls the function with SBS scale argument.</i>
--------------	--

Description

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

Usage

```
g_pkcg01_sbs(data, ...)
```

Arguments

data	Data to be passed into the plotting function.
...	Any other parameters to be passed into the plotting function.

Value

ggplot2 object for pkcg01.

g_pkcg02_lin	<i>Wrapper around aNCA::pkcg02() function. Calls the function with LIN scale argument.</i>
--------------	--

Description

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

Usage

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

Arguments

data	Data to be passed into the plotting function.
...	Any other parameters to be passed into the plotting function.

Value

ggplot2 object for pkcg02.

g_pkcg02_log	<i>Wrapper around aNCA::pkcg02() function. Calls the function with LOG scale argument.</i>
--------------	--

Description

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

Usage

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

Arguments

data	Data to be passed into the plotting function.
...	Any other parameters to be passed into the plotting function.

Value

ggplot2 object for pkcg02.

g_pkcg02_sbs	<i>Wrapper around aNCA::pkcg02() function. Calls the function with SBS scale argument.</i>
--------------	--

Description

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

Usage

```
g_pkcg02_sbs(data, ...)
```

Arguments

data	Data to be passed into the plotting function.
...	Any other parameters to be passed into the plotting function.

Value

ggplot2 object for pkcg02.

g_pkcg03_lin	<i>Wrapper around aNCA::pkcg03() function. Calls the function with LIN scale argument.</i>
--------------	--

Description

Wrapper around aNCA::pkcg03() function. Calls the function with LIN scale argument.

Usage

```
g_pkcg03_lin(data, ...)
```

Arguments

data	Data to be passed into the plotting function.
...	Any other parameters to be passed into the plotting function.

Value

ggplot2 object for pkcg03.

g_pkcg03_log	<i>Wrapper around aNCA::pkcg03() function. Calls the function with LOG scale argument.</i>
--------------	--

Description

Wrapper around aNCA::pkcg03() function. Calls the function with LOG scale argument.

Usage

```
g_pkcg03_log(data, ...)
```

Arguments

data	Data to be passed into the plotting function.
...	Any other parameters to be passed into the plotting function.

Value

ggplot2 object for pkcg03.

g_pkcg03_sbs	<i>Wrapper around aNCA::pkcg03() function. Calls the function with SBS scale argument.</i>
--------------	--

Description

Wrapper around aNCA::pkcg03() function. Calls the function with SBS scale argument.

Usage

```
g_pkcg03_sbs(data, ...)
```

Arguments

data	Data to be passed into the plotting function.
...	Any other parameters to be passed into the plotting function.

Value

ggplot2 object for pkcg03.

generate_pre_specs	<i>Generate Pre-Specifications for CDISC Datasets</i>
--------------------	---

Description

Extracts variable-level metadata from metadata_nca_variables for the requested CDISC datasets (ADNCA, ADPP, PP). When cdisc_data is provided, specs are narrowed to only the variables present in each dataset.

Usage

```
generate_pre_specs(datasets = c("ADNCA", "ADPP", "PP"), cdisc_data = NULL)
```

Arguments

datasets	Character vector of dataset names to generate specs for. Defaults to c("ADNCA", "ADPP", "PP").
cdisc_data	Optional named list of data frames (as returned by export_cdisc()). Names should be lowercase (pp, adpp, adnca). When provided, each spec is filtered to only variables present as columns in the corresponding data frame.

Value

A named list of data frames, one per dataset, each containing columns: Dataset, Order, Variable, Label, Type, Role, Core, Length.

generate_tooltip_text	<i>Generate HTML Tooltip Text</i>
-----------------------	-----------------------------------

Description

Generate HTML Tooltip Text

Usage

```
generate_tooltip_text(data, labels_df, tooltip_vars, type)
```

Arguments

data	A data.frame with the source data.
labels_df	A data.frame used by get_label() to find variable labels.
tooltip_vars	A character vector of column names to include in the tooltip.
type	A character string specifying the label type for get_label().

Details

Creates a character vector of HTML tooltips for each row of a data frame, suitable for use with ggplotly. The output vector of this function should be added to original plotting data as a column, which then can be used as tooltip argument in the plotting function.

Value

A character vector of formatted HTML tooltip strings.

Examples

```
# Sample data
my_data <- data.frame(
  USUBJID = c("Subject-01", "Subject-02"),
  DOSE = c(100, 200),
  RESPONSE = c(5.4, 8.1)
)

my_labels <- data.frame(
  Dataset = "ADNCA",
  Variable = "USUBJID",
  Label = "Unique Subject ID"
) # Dummy labels object

vars_to_show <- c("USUBJID", "DOSE", "RESPONSE")

# Generate the tooltip text vector
tooltips <- generate_tooltip_text(my_data, my_labels, vars_to_show, "ADNCA")
my_data$tooltip <- tooltips
```

get_conversion_factor *Transform Units*

Description

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

Usage

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

Arguments

initial_unit	A character string representing the initial unit.
target_unit	A character string representing the target unit.

Value

A numeric value for the conversion factor from the initial to the target unit, or NA if the units are not convertible.

Examples

```
get_conversion_factor("meter", "kilometer")
get_conversion_factor("sec", "min")
```

get_halflife_plots	Create a Plotly Half-life Plot
--------------------	--------------------------------

Description

Generates a plotly plot for NCA half-life visualization, with a fit line and scatter points.

Usage

```
get_halflife_plots(pknca_data, add_annotations = TRUE, title_vars = NULL)
```

Arguments

pknca_data	PKNCA data object
add_annotations	Logical, whether to add the subtitle annotation
title_vars	Character vector of additional column names to always include in the plot title and use as merge keys when joining concentration data with NCA results, even when they have only one unique level. Columns not present in the data are silently ignored.

Value

A list with plotly objects and data

get_label	Get the Label of a Heading
-----------	----------------------------

Description

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

Usage

```
get_label(variable, type = "ADNCA", labels_df = metadata_nca_variables)
```

Arguments

variable	The variable for which the label is to be retrieved.
type	The type of the dataset for which the label is to be retrieved.
labels_df	A data frame containing at least the columns "Variable", "Label", and "Dataset".

Value

The label of the heading if it exists in the labels file, otherwise the variable name.

Examples

```

LABELS <- data.frame(
  Variable = c("USUBJID", "AVAL"),
  Label = c("Unique Subject Identifier", "Analysis Value"),
  Dataset = c("ADNCA", "ADNCA")
)
get_label("USUBJID", "ADNCA", LABELS) # Returns "Unique Subject Identifier"
get_label("AGE", "ADNCA", LABELS) # Returns "AGE"

```

get_settings_code	<i>Generate a session script from a settings YAML file</i>
-------------------	--

Description

Reads a settings YAML file and generates an R script that can reproduce the session. Mapping, filters, ratio table, and units are all read from the YAML.

Usage

```

get_settings_code(
  settings_file_path,
  data_path,
  output_path = "settings_code.R",
  template_path = system.file("www/templates/script_template.R", package = "aNCA")
)

```

Arguments

settings_file_path	Path to the YAML file containing the settings.
data_path	Path to the data file to be referenced in the script.
output_path	Path to write the resulting script file.
template_path	Path to the R script template file. By default, uses the one installed from your aNCA package version.

Value

Invisibly returns the output_path.

interval_add_impute	<i>Add specified imputation methods to the intervals in a PKNCAdata or data.frame object.</i>
---------------------	---

Description

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

Usage

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

Arguments

data	A PKNCAdata object containing the intervals data frame, or a data frame of intervals.
target_impute	A character string specifying the imputation method to be added.
after	Numeric value specifying the index in which the imputation will be added (optional). First is 0, last Inf. If missing, the imputation method is added at the end (Inf).
target_params	A character vector specifying the parameters to be targeted (optional). If missing, all TRUE in the intervals are taken.
target_groups	A data frame specifying the intervals to be targeted (optional). If missing, all relevant groups are considered.
...	arguments passed to interval_add_impute.

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.

Value

A modified PKNCAdata object with specified imputation methods on the target intervals.

Examples

```

d_conc <- data.frame(
  conc = c(1, 0.6, 0.2, 0.1, 0.9, 0.4, 1.2, 0.8, 0.3, 0.2, 1.1, 0.5),
  time = rep(0:5, 2),
  ID = rep(1:2, each = 6),
  param = rep(c("Analyte1", "Analyte2"), each = 6)
)

d_dose <- data.frame(
  dose = c(100, 200),
  time = c(0, 0),
  ID = c(1, 2)
)

o_conc <- PKNCA::PKNCAconc(d_conc, conc ~ time | ID / param)
o_dose <- PKNCA::PKNCAdose(d_dose, dose ~ time | ID)

intervals <- data.frame(
  start = c(0, 0, 0),
  end = c(3, 5, Inf),
  half.life = c(TRUE, TRUE, TRUE),
  cmax = c(TRUE, TRUE, TRUE),
  impute = c("start_conc0,start_predose", "start_predose", "start_conc0"),
  param = c("Analyte1", "Analyte2", "Analyte1")
)

o_data <- PKNCA::PKNCAdata(o_conc, o_dose, intervals = intervals)

# Apply interval_add_impute function
o_data <- interval_add_impute(o_data,
                             target_impute = "start_conc0",
                             target_params = "half.life",
                             target_groups = data.frame(param = "Analyte1"))

```

interval_remove_impute

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

Description

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

Usage

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

Arguments

<code>data</code>	A PKNCAdata object containing the intervals data frame, or a data frame of intervals.
<code>target_impute</code>	A character string specifying the imputation method to remove.
<code>...</code>	arguments passed to <code>interval_remove_impute</code> .

Value

A modified object with the specified imputations removed from the targeted intervals.

Examples

```
d_conc <- data.frame(
  conc = c(1, 0.6, 0.2, 0.1, 0.9, 0.4, 1.2, 0.8, 0.3, 0.2, 1.1, 0.5),
  time = rep(0:5, 2),
  ID = rep(1:2, each = 6),
  param = rep(c("Analyte1", "Analyte2"), each = 6)
)

d_dose <- data.frame(
  dose = c(100, 200),
  time = c(0, 0),
  ID = c(1, 2)
)

o_conc <- PKNCA::PKNCAconc(d_conc, conc ~ time | ID / param)
o_dose <- PKNCA::PKNCAdose(d_dose, dose ~ time | ID)

intervals <- data.frame(
  start = c(0, 0, 0),
  end = c(3, 5, Inf),
  half.life = c(TRUE, FALSE, TRUE),
  cmax = c(TRUE, TRUE, TRUE),
  impute = c("start_conc0", "start_predose", "start_predose", "start_conc0"),
  param = c("Analyte1", "Analyte2", "Analyte1")
)

o_data <- PKNCA::PKNCAdata(o_conc, o_dose, intervals = intervals)

# Apply interval_remove_impute function
o_data <- interval_remove_impute(data = o_data,
  target_impute = "start_conc0",
  target_params = "half.life",
  target_groups = data.frame(param = "Analyte1"))
```

Description

Helper function for BLQ processing

Usage

```
keep_blq_timepoints(plot_data, xvar, mean_group_var)
```

Arguments

`plot_data` Data frame containing the data to be processed.
`xvar` Character string of the variable name for the x-axis.
`mean_group_var` Character string specifying the grouping variable to plot by group.

Value

A data frame with valid timepoints based on BLQ criteria.

l_pkcl01	<i>Create PK Concentration Listing</i>
----------	--

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.

Usage

```
l_pkcl01(
  data,
  listgroup_vars = c("PARAM", "PCSPEC", "ROUTE"),
  grouping_vars = c("TRT01A", "USUBJID", "ATPTREF"),
  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"
)
```

Arguments

`data` A data frame containing the PK concentration data.
`listgroup_vars` A character vector specifying the variables to separate lists.
`grouping_vars` A character vector specifying the grouping variables within each list.

<code>displaying_vars</code>	A character vector specifying the variables to display in the listing.
<code>formatting_vars_table</code>	A data frame with the formatting of each variable. See details.
<code>title</code>	A character string to parse specifying the main title for the entire listing.
<code>subtitle</code>	A character string to parse specifying the subtitle to use for each list.
<code>footnote</code>	A character string to parse specifying the footnote of the listing table.

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.

Value

A list of listings, each corresponding to a unique combination of the grouping variables.

Author(s)

Gerardo Rodriguez

Examples

```
# Create a sample dataframe 'data' with the required variables
set.seed(123)
data <- data.frame(
  PARAM = rep(c("Param1", "Param2"), each = 6),
  PCSPEC = rep(c("Blood", "Urine"), each = 6),
  TRT01A = rep(c("Treatment1", "Treatment2"), each = 6),
  USUBJID = rep(c(rep(1, 3), rep(2, 3)), 2),
  NFRLT = rep(1:3, 4),
  AFRLT = rep(1:3, 4) + runif(12, 0, 0.5),
  TIMEU = "hours",
  AVAL = rep(0:2, 4) + runif(12, 0, 0.5),
  AVALU = "mg/L"
```

```

)

# Define the formatting table
formatting_vars_table <- data.frame(var_name = names(data),
                                     Label = c("Parameter", "Specimen", "Treatment Arm",
                                                "Unique Subject ID", "Norminal Time ($TIMEU)",
                                                "Actual Time ($TIMEU)", "Time Unit",
                                                "Analyte Value ($AVALU)", "Analyte Unit"),
                                     na_str = "Missing",
                                     zero_str = c(rep("0", 7), "BLQ", "0"),
                                     align = "center",
                                     format_fun = c(NA, NA, NA, NA,
                                                    "round", "round", NA, "round", NA),
                                     digits = c(NA, NA, NA, NA, 2, 2, NA, 3, NA))

# Call the l_pkcl01 function with the sample data
listing_ex <- l_pkcl01(data = data,
                      listgroup_vars = c("PARAM", "PCSPEC"),
                      grouping_vars = c("TRT01A", "USUBJID"),
                      displaying_vars = c("NFRLT", "AFRLT", "AVAL"),
                      formatting_vars_table = formatting_vars_table,
                      title = "Listing of PK Concentration",
                      subtitle = "Subjects with !PARAM: $PARAM (!PCSPEC: $PCSPEC)"
)

print(listing_ex)

```

l_pkcl01_tad

Wrapper around aNCA::l_pkcl01() for TAD-based concentration listings.

Description

Wrapper around aNCA::l_pkcl01() for TAD-based concentration listings.

Usage

```
l_pkcl01_tad(data, ...)
```

Arguments

data	Data to be passed into the listing function.
...	Any other parameters to be passed into the listing function.

Value

A named list of listing_df objects.

l_pkcl02_uri

*Urine Concentration and Volume Listing (pkcl02)***Description**

Filters ADNCA to urine specimen rows (where PCSPEC %in% urine_specs) then delegates to [l_pkcl01\(\)](#) with VOLUME and VOLUMEU added to the displayed columns when those columns are present in the data.

Usage

```
l_pkcl02_uri(
  data,
  urine_specs = c("URINE"),
  listgroup_vars = c("PARAM", "PCSPEC"),
  displaying_vars = NULL,
  ...
)
```

Arguments

data	A CDISC ADNCA data frame (from <code>export_cdisc()</code> \$adnca).
urine_specs	Character vector of specimen type values to keep, matched case-insensitively against PCSPEC. Default: <code>c("URINE")</code> .
listgroup_vars	Character vector of columns used to split output into separate listings. Default: <code>c("PARAM", "PCSPEC")</code> . When PCSPEC is absent from data, it is silently removed from this vector.
displaying_vars	Character vector of columns to display. When NULL (default), uses <code>c("NFRLT", "AFRLT", "AVAL")</code> plus VOLUME/VOLUMEU if those columns exist in the data.
...	Additional arguments forwarded to l_pkcl01() .

Value

A named list of `listing_df` objects.

Examples

```
## Not run:
adnca <- export_cdisc(res_nca)$adnca
listings <- l_pkcl02_uri(adnca)
print(listings[[1]])

## End(Not run)
```

l_pkpl01

*Individual PK Parameters Listing (pkpl01)***Description**

Creates individual-level listings of PK parameters from ADPP data using the same engine as `l_pkcl01()`. Returns one listing per unique combination of `listgroup_vars` (default: PPCAT x PPSPEC).

Usage

```
l_pkpl01(
  data,
  listgroup_vars = c("PPCAT", "PPSPEC"),
  grouping_vars = c("TRT01A", "USUBJID"),
  param_var = "PARAM",
  value_var = "AVAL",
  unit_var = "AVALU",
  title = "Listing of Individual PK Parameters",
  subtitle = NULL,
  footnote = NULL
)

l_pkpl01_mp(data, ...)
```

Arguments

<code>data</code>	A CDISC ADPP data frame (from <code>export_cdisc()</code> \$adpp).
<code>listgroup_vars</code>	Character vector of columns used to split the output into separate listings. Default: <code>c("PPCAT", "PPSPEC")</code> .
<code>grouping_vars</code>	Character vector of key/header columns within each listing (shown as indented row keys). Default: <code>c("TRT01A", "USUBJID")</code> .
<code>param_var</code>	Column whose unique values become display columns after pivoting wide. Default: <code>"PARAM"</code> .
<code>value_var</code>	Column containing the numeric analysis value. Default: <code>"AVAL"</code> .
<code>unit_var</code>	Column containing the parameter unit used to build column headers (" <code><PARAM></code> (<code><unit></code>)"). Default: <code>"AVALU"</code> .
<code>title</code>	Main listing title string.
<code>subtitle</code>	Per-listing subtitle. Supports <code>\$VAR / !VAR</code> annotation syntax. Defaults to the unique values of <code>listgroup_vars</code> .
<code>footnote</code>	Footnote string.
<code>...</code>	Additional arguments forwarded to <code>l_pkpl01()</code> .

Value

A named list of `listing_df` objects (one per `listgroup_vars` combination), suitable for printing in a Shiny `verbatimTextOutput`.

Functions

- `l_pkpl01_mp()`: Listing filtered to metabolite rows (pkpl01 M/P). Uses the same METABFL -> PPCAT -> PARAM fallback as `t_pkpt03_MP_col()`.

Examples

```
## Not run:
adpp <- export_cdisc(res_nca)$adpp
listings <- l_pkpl01(adpp)
print(listings[[1]])

## End(Not run)
```

<code>l_pkpl04_mp</code>	<i>Individual Treatment Comparison Listing (pkpl04)</i>
--------------------------	---

Description

Produces a per-subject listing of individual PK parameter values organised for treatment comparison. Each listing page covers one PPCAT/PPSPEC combination. PK parameters become display columns (one column per PARAM value) and the rows are keyed by TRT01A and USUBJID.

Usage

```
l_pkpl04_mp(
  data,
  grouping_vars = c("PARAM", "TRT01A", "USUBJID"),
  title = "Listing of Individual PK Parameter Values by Treatment",
  ...
)
```

Arguments

<code>data</code>	A CDISC ADPP data frame (from <code>export_cdisc()</code> \$adpp).
<code>grouping_vars</code>	Columns used to identify row keys before pivoting. PARAM must be included so it is spread into display columns. Default: <code>c("PARAM", "TRT01A", "USUBJID")</code> .
<code>title</code>	Main listing title string.
<code>...</code>	Additional arguments forwarded to <code>l_pkpl01()</code> .

Details

This listing shows the raw individual AVAL values from ADPP, not pre-computed ratios. If your ADPP contains NCA ratio parameters (e.g. metabolite-to-parent AUC ratios added via the aNCA ratio-calculation module), those parameters are displayed here just like any other PARAM row. The `_mp` suffix in the function name reflects its typical use with metabolite/parent ratio parameters, but no metabolite filtering is applied – all PARAM values in the data are included.

Note: PARAM is listed in grouping_vars so that it participates in the pivot_wider step (each unique PARAM value becomes a column header). After pivoting, PARAM is no longer a row-key column; the actual listing keys are TRT01A and USUBJID.

Value

A named list of listing_df objects.

Examples

```
## Not run:
adpp <- export_cdisc(res_nca)$adpp
listings <- l_pkpl04_mp(adpp)
print(listings[[1]])

## End(Not run)
```

metadata_nca_parameters

metadata_nca_parameters

Description

A dataset containing the mapping between PKNCA terms and CDISC terms. It mainly involves:

- Parameters: PPTEST, PPTESTCD

Usage

```
metadata_nca_parameters
```

Format

A data frame with 123 rows and 6 variables:

PKNCA PKNCA term

PPANMETH PPANMETH term, specifying when relevant the calculation method for PPTESTCD

PPTESTCD CDISC term

PPTEST Official CDISC term

input_names Combination of PPTESTCD + ": " + PPTEST. Used for App inputs

FUN PKNCA function used to calculate the parameter

description PKNCA description of the term

is_cdisc_sure Logical indicating if the term is a CDISC official name

unit_type Type of unit associated with the term

TYPE Type of data associated with the parameter/term

CAT Arbitrary assigned subclass for the parameter/term

Depends PKNCA derived. Designates all directly used parameters in calculation

can_excretion Logical. Indicates if the parameter can be used in excretion analysis

can_non_excretion Logical. Indicates if the parameter can be used in non-excretion analysis

can_single_dose Logical. Indicates if the parameter can be used in single dose analysis

can_multiple_dose Logical. Indicates if the parameter can be used in multiple dose analysis

can_extravascular Logical. Indicates if the parameter can be used in extravascular analysis

can_metabolite Logical. Indicates if the parameter can be used in metabolite analysis

show_in_matrix_selection Logical. Indicates if the parameter will be shown for general interval calculations

Source

Generated for use in the translate_nomenclature function.

metadata_nca_variables

metadata_nca_variables

Description

A dataset containing pharmacokinetic variable specifications.

Usage

metadata_nca_variables

Format

A data frame with 366 rows and 19 variables:

Dataset Character. Indicates the dataset the variable belongs to (PP, ADNCA, ADPP).

Order Numeric. Variable order within its domain, based on Role, Core and Variable

Variable Character. The short name of the variable.

Label Character. A descriptive label for the variable.

Type Character. Data type of the variable (Char, Num, text, integer, float, dateTime).

Role Character. The CDISC role of the variable (e.g., Identifier, Topic, Timing...).

Core Character. Indicates the core status of the variable (Req = Required, Perm = Permissible, Exp = Expected, Cond = Conditional).

company_specific Logical. Indicates if the variable is company-specific (not CDISC).

is.core Logical. TRUE if the variable is a core variable (always needed to be present).

Length Numeric. The maximum length of the variable.

Controlled_Terms Character. Reference to controlled terminology (e.g., C85839, C66731).

is.used Logical. TRUE if the variable is meant to be included.

Values Character. Possible values (if applicable) for the variable separated by ', '.

is.mapped Logical. TRUE if the variable is mapped in ADNCA (App's input).

mapping_tooltip Character. Tooltip text for mapping guidance in the App.

mapping_section Character. Mapping section where the variable is classified in the App.

mapping_alternatives Character. Alternative column names for the variable.

mapping_order Numeric. Defines the mapped variables order in the mapped dataset

allow_create_numeric Logical. Indicates if the mapping input should allow the user to introduce custom numbers

Source

Used for PP and ADPP mapping rules and checks in the export_cdisc function

multiple_matrix_ratios

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.

Description

The ratios are calculated as specimen1 / specimen 2.

Usage

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

Arguments

data	A data frame containing the concentration data.
matrix_col	A character string specifying the column name for the matrix type.
conc_col	A character string specifying the column name for the concentration data.
units_col	A character string specifying the column name for the units.

groups	A character vector of grouping variables to use for the analysis. Must include time column, USUBJID, and optionally, other grouping variables.
spec1	A character string specifying the value for the first specimen type(s) in the matrix_col.
spec2	A character string specifying the value for the second specimen type(s) in the matrix_col.

Value

A data frame containing the ratios.

Examples

```
data <- data.frame(
  USUBJID = c("A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "A", "A"),
  NFRLT = c(0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2),
  MATRIX = c(
    "BLOOD", "BLOOD", "BLOOD", "PLASMA", "PLASMA", "PLASMA",
    "BRAIN", "BRAIN", "BRAIN", "LIVER", "LIVER", "LIVER"
  ),
  CONC = c(10, 20, 15, 25, 30, 40, 5, 10, 8, 12, 18, 16),
  UNITS = rep("ng/mL", 12)
)
multiple_matrix_ratios(data, "MATRIX", "CONC", "UNITS", c("NFRLT", "USUBJID"), "BLOOD", "PLASMA")
```

p_pkcg03_lin_dose	<i>Wrapper around aNCA::pkcg03() function. Mean linear plot grouped by dose.</i>
-------------------	--

Description

Wrapper around aNCA::pkcg03() function. Mean linear plot grouped by dose.

Usage

```
p_pkcg03_lin_dose(data, ...)
```

Arguments

data	Data to be passed into the plotting function.
...	Any other parameters to be passed into the plotting function.

Value

ggplot2 object for pkcg03.

p_pkcg03_log_dose	<i>Wrapper around aNCA::pkcg03() function. Mean log plot grouped by dose.</i>
-------------------	---

Description

Wrapper around aNCA::pkcg03() function. Mean log plot grouped by dose.

Usage

```
p_pkcg03_log_dose(data, ...)
```

Arguments

data	Data to be passed into the plotting function.
...	Any other parameters to be passed into the plotting function.

Value

ggplot2 object for pkcg03.

p_pkcg03_sbs_dose	<i>Wrapper around aNCA::pkcg03() function. Mean side-by-side plot grouped by dose.</i>
-------------------	--

Description

Wrapper around aNCA::pkcg03() function. Mean side-by-side plot grouped by dose.

Usage

```
p_pkcg03_sbs_dose(data, ...)
```

Arguments

data	Data to be passed into the plotting function.
...	Any other parameters to be passed into the plotting function.

Value

ggplot2 object for pkcg03.

p_pkpg01_cum

*Mean Urine PK Parameter Profile Plot (pkpg01)***Description**

Computes mean ($\pm$ SD) of a urine PK parameter per treatment arm across collection intervals and draws a connected line plot. Designed for ADPP data filtered to PPSPEC %in% urine_specs (e.g. cumulative amount excreted or fraction of dose recovered). Returns one ggplot per unique combination of list_vars.

Usage

```
p_pkpg01_cum(
  data,
  strat_var = "TRT01A",
  param_var = "PARAM",
  value_var = "AVAL",
  urine_specs = c("URINE"),
  paramcd_filter = "RCAMINT",
  time_end_var = "PPENINT",
  list_vars = c("PPCAT"),
  title = "Mean Cumulative Amount Recovered in Urine",
  subtitle = NULL,
  footnote = NULL,
  xlab = NULL,
  ylab = NULL
)

p_pkpg01_per(
  data,
  paramcd_filter = "FREXINT",
  title = "Mean Percentage of Dose Recovered in Urine",
  ylab = "Percent Dose Recovered (%)",
  ...
)
```

Arguments

data	A CDISC ADPP data frame (from export_cdisc())\$adpp).
strat_var	Column for treatment arm colour/grouping. Default: "TRT01A".
param_var	Column whose unique values label each x-axis position. Default: "PARAM".
value_var	Column containing the numeric analysis value. Default: "AVAL".
urine_specs	Character vector of PPSPEC values treated as urine, matched case-insensitively. Default: c("URINE").

paramcd_filter	Character vector of PARAMCD values to keep. Only rows with a matching PARAMCD are plotted. Pass NULL to skip the filter. Default: "RCAMINT" (cumulative amount recovered per interval).
time_end_var	Column containing the collection interval end time (ISO 8601 duration string or numeric hours). Used as a numeric x-axis when parseable; falls back to param_var labels otherwise. Default: "PPENINT".
list_vars	Columns used to split output into separate plots. Default: c("PPCAT").
title	Optional plot title.
subtitle	Optional plot subtitle.
footnote	Optional footnote / caption.
xlab	X-axis label. Default: "Collection Interval".
ylab	Y-axis label. Defaults to the label attribute of value_var.
...	Additional arguments forwarded to p_pkpg01_cum() .

Value

A named list of ggplot objects.

Functions

- [p_pkpg01_per\(\)](#): Mean percentage of dose recovered in urine (pkpg01 %). Identical to [p_pkpg01_cum\(\)](#) but defaults to a % dose recovered title and y-axis label.

Examples

```
## Not run:
adpp <- export_cdisc(res_nca)$adpp
plots <- p_pkpg01_cum(adpp)
plots[[1]]

## End(Not run)
```

p_pkpg02_doseprop	<i>Dose-Proportionality Scatter Plot with Power-Model Regression (pkpg02)</i>
-------------------	---

Description

Plots individual AVAL values against dose (DOSEA) on a log-log scale with one facet per PK parameter. A power-model regression line ($\log y = a + b * \log x$) is overlaid on each facet together with the slope estimate and its confidence interval, enabling visual assessment of dose-proportionality (slope $b = 1$).

Usage

```
p_pkpg02_doseprop(
  data,
  dose_var = "DOSEA",
  value_var = "AVAL",
  param_var = "PARAM",
  strat_var = "TRT01A",
  list_vars = c("PPCAT", "PPSPEC"),
  ci_level = 0.9,
  log_scale = TRUE,
  title = NULL,
  subtitle = NULL,
  footnote = NULL,
  xlab = NULL,
  ylab = NULL
)
```

Arguments

data	A CDISC ADPP data frame (from <code>export_cdisc()</code> \$adpp).
dose_var	Column containing the administered dose. Default: "DOSEA".
value_var	Column containing the PK parameter value. Default: "AVAL".
param_var	Column used as facet variable. Default: "PARAM".
strat_var	Column used for point colour. Default: "TRT01A".
list_vars	Columns used to split output into separate plots. Default: c("PPCAT", "PPSPEC").
ci_level	Confidence level for the slope CI. Default: 0.90.
log_scale	Logical. When TRUE (default), both axes are log10-scaled.
title	Optional plot title.
subtitle	Optional plot subtitle.
footnote	Optional footnote / caption.
xlab	X-axis label. Defaults to dose_var label + unit.
ylab	Y-axis label. Defaults to the label attribute of value_var.

Value

A named list of ggplot objects.

Examples

```
## Not run:
adpp <- export_cdisc(res_nca)$adpp
plots <- p_pkpg02_doseprop(adpp)
plots[[1]]

## End(Not run)
```

p_pkpg03_boxp

*Boxplot of Primary PK Parameters (pkpg03)***Description**

Produces one boxplot per PK parameter with treatment arms on the x-axis. Returns a named list of ggplot objects, one per PPCAT x PPSPEC combination in the ADPP data.

Usage

```
p_pkpg03_boxp(
  data,
  strat_var = "TRT01A",
  param_var = "PARAM",
  value_var = "AVAL",
  list_vars = c("PPCAT", "PPSPEC"),
  all_points = FALSE,
  title = NULL,
  subtitle = NULL,
  footnote = NULL,
  ylab = NULL
)

p_pkpg04_boxp(data, ...)
```

Arguments

data	A CDISC ADPP data frame (from <code>export_cdisc()</code> \$adpp).
strat_var	Column used for x-axis grouping (treatment arms). Default: "TRT01A".
param_var	Column whose unique values each become a separate plot. Default: "PARAM".
value_var	Column containing the numeric analysis value. Default: "AVAL".
list_vars	Columns used to split output into separate plot pages. Default: c("PPCAT", "PPSPEC").
all_points	Logical. When TRUE, individual data points are overlaid on the boxes (pkpg04 style). Default: FALSE.
title	Optional plot title string.
subtitle	Optional plot subtitle string.
footnote	Optional footnote string.
ylab	Y-axis label. Defaults to the label attribute of value_var.
...	Additional arguments forwarded to p_pkpg03_boxp() .

Value

A named list of ggplot objects.

Functions

- `p_pkpg04_boxp()`: Boxplot with all individual data points overlaid (pkpg04).

Examples

```
## Not run:
adpp <- export_cdisc(res_nca)$adpp
plots <- p_pkpg03_boxp(adpp)
plots[[1]]

## End(Not run)
```

p_pkpg06_mp

Boxplot of Metabolite/Parent PK Parameter Ratios (pkpg06)

Description

Filters ADPP to metabolite rows using the same fallback logic as `t_pkpt03_MP_col()` (METABFL preferred, then PPCAT/PARAM grep for "metab"), then delegates to `p_pkpg03_boxp()`.

Usage

```
p_pkpg06_mp(data, ...)
```

Arguments

<code>data</code>	A CDISC ADPP data frame (from <code>export_cdisc()</code> \$adpp).
<code>...</code>	Additional arguments forwarded to <code>p_pkpg03_boxp()</code> .

Value

A named list of ggplot objects (same format as `p_pkpg03_boxp()`).

Examples

```
## Not run:
adpp <- export_cdisc(res_nca)$adpp
plots <- p_pkpg06_mp(adpp)
plots[[1]]

## End(Not run)
```

parse_annotation	Generate an Empty Plotly Object
------------------	---------------------------------

Description

This function returns a blank Plotly plot with optional annotation text. It ensures that when no valid data is available, a meaningful placeholder plot is displayed instead of causing an error.

Usage

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

Arguments

data	Data frame containing data to reference. Should include columns and labels referenced in the text string. Referenced variables should be able to produce single unique result.
text	Character text to parse.

Details

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).

- `\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.

Value

Parsed annotation text.

pc_example	<i>pc_example</i>
------------	-------------------

Description

An SDTM PC-like pharmacokinetic concentrations dataset derived from [adnca_example](#). Intended to support and test the PC+EX upload workflow (see GitHub issue #624).

Usage

```
pc_example
```

Format

A data frame with the same number of rows as `adnca_example` and 11 variables:

DOMAIN Character. Domain Abbreviation.

STUDYID Character. Study identifier.

USUBJID Character. Unique subject identifier.

PCTEST Character. Name of the measured analyte (mapped from PARAM).

PCSPEC Character. Specimen type (e.g., SERUM, URINE).

PCSTRESN Numeric. Numeric concentration result (mapped from AVAL).

PCSTRESU Character. Concentration units (mapped from AVALU).

PCDTC Character. Sample collection datetime (ISO 8601), synthesised from AFRLT using an arbitrary reference origin.

PCRFTDTC Character. Reference dose datetime (ISO 8601), synthesised from AFRLT - ARRLT.

PCELTm Character. Planned elapsed time from reference dose in ISO 8601 duration format (e.g., "PT0.5H"), derived from NFRLT.

VOLUME Numeric. Volume of specimen collected (mL), for urine samples.

VOLUMEU Character. Volume units.

Source

Derived from `adnca_example` and `dm_example` via `data-raw/sdtm_example.R`.

`pivot_wider_pknca_results`

Reshape PKNCA Results

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 added as additional columns.

Usage

```
pivot_wider_pknca_results(myres, flag_rules = NULL, extra_vars_to_keep = NULL)
```

Arguments

myres	The output of PKNCA::pk.nca. It makes some additional assumptions: 1. CDISC denomination of actual and nominal time variables (AFRLT, ARRLT, NFRLT, NRRLT). 2. Intervals must include a column (type_interval) to differentiate between the custom AUC ranges ("manual") and main parameter calculations ("main"). 3. Includes PPSTRES and PPSTRESU variables in results dataset. 4. Columns start_dose and end_dose must express the actual start and end times of the dose, relative to the last reference dose. 5. Temporarily: CDISC denomination of PK parameters related to half-life: "LAMZNPT", "LAMZLL", "LAMZ" Used to derive LAMZNPT and LAMZMTD.
flag_rules	A named list of flagging rules to be applied to the results. Each rule should be a list with two elements: is.checked (logical) indicating whether the rule should be applied, and threshold (numeric) specifying the threshold value for flagging. The name of each rule should correspond to a parameter in the results data.frame as a PPTTESTCD (e.g., "R2ADJ", "AUCPEO", "AUCPEP", "LAMZSPN").
extra_vars_to_keep	Optional character vector of variable names to join from the concentration data to the output. Default is NULL.

Value

A data frame which provides an easy overview on the results from the NCA in each profile/subject and how it was computed lambda (half life) and the results of the NCA parameters (cmax, AUC, AUClast), including new columns Exclude (a derivation from pknc\$aexclude), Missing (indicating if flag parameters are missing from PKNCA calculation), and flagged (indicating if the row is ACCEPTED, FLAGGED or MISSING based on the flagging rules).

pk.calc.ermax	<i>Calculate the maximum excretion rate</i>
---------------	---

Description

Calculate the maximum excretion rate

Usage

```
pk.calc.ermax(conc, volume, time, duration.conc, check = TRUE)
```

Arguments

conc	The concentration in the excreta (e.g., urine or feces)
volume	The volume (or mass) of the sample
time	The starting time of the collection interval
duration.conc	The duration of the collection interval
check	Should the concentration data be checked?

Value

The maximum excretion rate, or NA if not available

pk.calc.ertlst	<i>Calculate the midpoint collection time of the last measurable excretion rate</i>
----------------	---

Description

Calculate the midpoint collection time of the last measurable excretion rate

Usage

```
pk.calc.ertlst(conc, volume, time, duration.conc, check = TRUE)
```

Arguments

conc	The concentration in the excreta (e.g., urine or feces)
volume	The volume (or mass) of the sample
time	The starting time of the collection interval
duration.conc	The duration of the collection interval
check	Should the concentration and time data be checked?

Value

The midpoint collection time of the last measurable excretion rate, or NA/0 if not available

pk.calc.ertmax	<i>Calculate the midpoint collection time of the maximum excretion rate</i>
----------------	---

Description

Calculate the midpoint collection time of the maximum excretion rate

Usage

```
pk.calc.ertmax(
  conc,
  volume,
  time,
  duration.conc,
  check = TRUE,
  first.tmax = NULL,
  options = list()
)
```

Arguments

conc	The concentration in the excreta (e.g., urine or feces)
volume	The volume (or mass) of the sample
time	The starting time of the collection interval
duration.conc	The duration of the collection interval
check	Should the concentration and time data be checked?
first.tmax	If TRUE, return the first time of maximum excretion rate; otherwise, return the last
options	List of PKNCA global options set

Value

The midpoint collection time of the maximum excretion rate, or NA if not available

pk.calc.volpk	<i>Calculate the total urine volume</i>
---------------	---

Description

Calculate the total urine volume

Usage

```
pk.calc.volpk(volume)
```

Arguments

volume	The volume (or mass) of the sample
--------	------------------------------------

Value

The sum of urine volumes for the interval

pk_dose_qc_plot

*Create a PK Dose Quality Control (QC) Plot***Description**

Generates a PK Dose QC plot by layering concentration data (as black shapes) and dose data (as colored points). It creates a single, unified legend for both data types and can return either a static ggplot or an interactive plotly object.

Usage

```
pk_dose_qc_plot(
  data_conc,
  data_dose = NULL,
  x_var,
  y_var,
  colour_var,
  shape_var,
  grouping_vars,
  other_tooltip_vars = NULL,
  x_var_units = NULL,
  colour_var_units = NULL,
  labels_df = metadata_nca_variables,
  title = NULL,
  show_pk_samples = TRUE,
  show_doses = TRUE,
  as_plotly = FALSE,
  height = NULL
)
```

Arguments

data_conc	A data.frame containing concentration data (e.g., PK samples).
data_dose	An optional data.frame containing dosing information.
x_var	Character. The column name to be used for the x-axis.
y_var	Character. The column name to be used for the y-axis.
colour_var	Character. The column in data_dose to map to color.
shape_var	Character. The column in data_conc to map to shape.
grouping_vars	Character vector. Column names to use for faceting.
other_tooltip_vars	Optional character vector of additional column names to include in the tooltip.
x_var_units	Character. The column name containing the units for the x-axis variable. It is expected that this column contains a single unique value.

colour_var_units	Character. The column name for the units of the colour variable in data_dose. It is expected that this column contains a single unique value.
labels_df	A data.frame used by helper functions to look up variable labels. It uses meta-data_nca_variables as default
title	Character. The main title for the plot.
show_pk_samples	Logical. If TRUE, plots the concentration data.
show_doses	Logical. If TRUE, plots the dose data.
as_plotly	Logical. If TRUE, converts the final plot to an interactive plotly object.
height	Numeric. Desired height for the plot.

Details

Unless specified, the variables required as arguments are expected to be present in both data_conc and data_dose.

Value

A ggplot object or, if as_plotly = TRUE, a plotly object.

Examples

```
# Sample concentration data
conc_data <- data.frame(
  USUBJID = rep(paste0("S-", 1:2), each = 2),
  ACTUAL_TIME = c(0, 24, 0, 24),
  SAMPLE_TYPE = rep(c("PLASMA", "URINE"), 2),
  COHORT = "A",
  TIME_UNIT = "hr"
)

# Sample dose data
dose_data <- data.frame(
  USUBJID = rep(paste0("S-", 1:2), each = 1),
  ACTUAL_TIME = c(0, 0),
  DOSE_LEVEL = c(100, 100),
  COHORT = "A",
  DOSE_UNIT = "mg"
)

# Generate the plot
pk_dose_qc_plot(
  data_conc = conc_data,
  data_dose = dose_data,
  x_var = "ACTUAL_TIME",
  y_var = "USUBJID",
  colour_var = "DOSE_LEVEL",
  shape_var = "SAMPLE_TYPE",
  grouping_vars = "COHORT",
```

```

x_var_units = "TIME_UNIT",
colour_var_units = "DOSE_UNIT",
title = "Sample Dosing and PK Plot"
)

```

pkcg01

Generate PK Concentration-Time Profile Plots

Description

This function generates a list of ggplots for PK concentration-time profiles. Records flagged with PKSUMXF == "Y" remain visible as crosses; other records retain circular points. A footnote explains the crosses in plots containing flagged records, including interactive plots and static exports.

Usage

```

pkcg01(
  adnca = 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
)

```

Arguments

adnca A data frame containing the data.

xvar	A character string of the variable name for the x-axis.
yvar	A character string of the variable name for the y-axis.
xvar_unit	A character string of the unit for the x-axis variable.
yvar_unit	A character string of the unit for the y-axis variable.
color_var	A character string of the variable name for the color.
color_var_label	A character string of the color label.
xbreaks_var	A character string of the x-axis breaks.
xbreaks_mindist	A numeric value for xbreak_mindist.
xmin	A numeric value for the minimum x-axis limit.
xmax	A numeric value for the maximum x-axis limit.
ymin	A numeric value for the minimum y-axis limit.
ymax	A numeric value for the maximum y-axis limit.
xlab	Character for x-axis label. Defaults: xvar label & xvar_unit.
ylab	Character for y-axis label. Defaults: yvar label & yvar_unit.
title	Character for plot title.
subtitle	Character for plot subtitle.
footnote	A character string of a manual footnote for the plot.
plotgroup_vars	A character vector of the variables to group data.
plotgroup_names	A character vector of the grouping variable names.
scale	Scale for the Y axis, either "LIN" or "LOG".
studyid	A character string specifying the study ID variable.
trt_var	A character string specifying the treatment variable.
plotly	Logical indicating whether to return plotly objects. Defaults to TRUE.

Value

A list of ggplot or plotly objects for each unique group.

Author(s)

Gerardo Rodriguez

Examples

```
adnca <- adnca_example
adnca <- subset(adnca, adnca$USUBJID %in% unique(adnca$USUBJID)[c(1, 2)])
attr(adnca[["AFRLT"]], "label") <- "Actual time from first dose"
attr(adnca[["AVAL"]], "label") <- "Analysis val"

plots_lin <- pkcg01(adnca = adnca, xmax = 1, scale = "LIN")
```

pkcg02

*Generate Combined PK Concentration-Time Profile Plot by Cohort***Description**

This function generates a list of plotly objects PK concentration-time profiles by group. Records flagged with `PKSUMXF == "Y"` remain visible as crosses; other records retain circular points. A footnote explains the crosses in plots containing flagged records, including interactive plots and static exports.

Usage

```
pkcg02(
  adnca = 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", USUBJID = "Subject ID"),
  scale = c("LIN", "LOG", "SBS")[1],
  studyid = "STUDYID",
  trt_var = "TRT01A",
  plotly = TRUE
)
```

Arguments

<code>adnca</code>	A data frame containing the data.
<code>xvar</code>	A character string of the variable name for the x-axis.
<code>yvar</code>	A character string of the variable name for the y-axis.
<code>xvar_unit</code>	A character string of the unit for the x-axis variable.

yvar_unit	A character string of the unit for the y-axis variable.
color_var	A character string of the variable name for the color.
color_var_label	A character string of the color label.
xbreaks_var	A character string of the x-axis breaks.
xbreaks_mindist	A numeric value for xbreak_mindist.
xmin	A numeric value for the minimum x-axis limit.
xmax	A numeric value for the maximum x-axis limit.
ymin	A numeric value for the minimum y-axis limit.
ymax	A numeric value for the maximum y-axis limit.
xlab	Character for x-axis label. Defaults: xvar label & xvar_unit.
ylab	Character for y-axis label. Defaults: yvar label & yvar_unit.
title	Character for plot title.
subtitle	Character for plot subtitle.
footnote	A character string of a manual footnote for the plot.
plotgroup_vars	A character vector of the variables to group data.
plotgroup_names	A character vector of the grouping variable names.
scale	Scale for the Y axis, either "LIN" or "LOG".
studyid	A character string specifying the study ID variable.
trt_var	A character string specifying the treatment variable.
plotly	Logical indicating whether to return plotly objects. Defaults to TRUE.

Value

A list of ggplot or plotly objects for each unique group.

Author(s)

Kezia Kobana

Examples

```
# Make an example small dataset
adnca <- adnca_example
adnca <- adnca[adnca$USUBJID %in% unique(adnca$USUBJID)[c(1, 2)],]
attr(adnca[["AFRLT"]], "label") <- "Actual time from first dose"
attr(adnca[["AVAL"]], "label") <- "Analysis value"

# Run the function
plots <- pkcg02(adnca)
plots_log <- pkcg02(adnca, scale = "LOG")
plotly::plotly_build(plots[[1]]) # View the first plot
```

pkcg03

*Generate PK Concentration-Time Profile Plots***Description**

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

Usage

```
pkcg03(
  adnca = data(),
  xvar = "NFRLT",
  yvar = "AVAL",
  xvar_unit = "RRLTU",
  yvar_unit = "AVALU",
  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"),
  plotgroup_names = list(ROUTE = "Route", PCSPEC = "Specimen", PARAM = "Analyte"),
  scale = c("LIN", "LOG", "SBS")[1],
  studyid = "STUDYID",
  mean_group_var = "DOSEA",
  plotly = TRUE,
  summary_method = "Mean_sdi",
  whiskers_lwr_upr = c("Both", "Upper", "Lower")[1]
)
```

Arguments

adnca	A data frame containing the data.
xvar	A character string of the variable name for the x-axis.
yvar	A character string of the variable name for the y-axis.
xvar_unit	A character string of the unit for the x-axis variable.
yvar_unit	A character string of the unit for the y-axis variable.
xbreaks_var	A character string of the x-axis breaks.
xbreaks_mindist	A numeric value for minimum distance between x-axis breaks in centimeters.

xmin	A numeric value specifying the minimum x-axis limit.
xmax	A numeric value specifying the maximum x-axis limit.
ymin	A numeric value for the minimum y-axis limit.
ymax	A numeric value for the maximum y-axis limit.
xlab	Character for x-axis label. Defaults: xvar label & xvar_unit.
ylab	Character for y-axis label. Defaults: yvar label & yvar_unit.
title	A character string for plot title.
subtitle	A character string for plot subtitle.
footnote	A character string of a manual footnote for the plot.
plotgroup_vars	A character vector of the variables to group data.
plotgroup_names	A character vector of the grouping variable names.
scale	A character string: "LIN", "LOG", or "SBS", defining the y-axis scale.
studyid	A character string specifying the study ID variable.
mean_group_var	A character string specifying the grouping variable to plot by group.
plotly	Logical indicating whether to return plotly objects. Defaults to TRUE.
summary_method	A character string specifying the stat method to summarize observations.
whiskers_lwr_upr	A character string specifying the whisker type (upper, lower or both)

Value

A list of ggplot objects for each unique group.

Author(s)

Kezia Kobana magic numbers for footnote position and margin, work in app up to 4 lines

Examples

```
## Not run:
adnca <- adnca_example
attr(adnca[["AFRLT"]], "label") <- "Actual time from first dose"
attr(adnca[["AVAL"]], "label") <- "Analysis val"

plots_lin <- pckg03(adnca = adnca, xmax = 1)
plots_log <- pckg03(adnca = adnca, scale = "LOG")
plots_sbs <- pckg03(
  adnca = adnca,
  xbreaks_var = "NFRLT",
  xmin = 100,
  xmax = 1000,
  scale = "SBS",
  plotly = TRUE,
  whiskers_lwr_upr = "Both"
)

## End(Not run)
```

PKNCA_build_units_table

Build Units Table for PKNCA

Description

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

Usage

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

Arguments

<code>o_conc</code>	A PKNCA concentration object (PKNCAconc).
<code>o_dose</code>	A PKNCA dose object (PKNCAdose).

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.

Any NA units in groups already containing at least one valid value will be ignored from the creation of the units table.

Value

A data frame containing the PKNCA formatted units table.

Examples

```
# Assuming `o_conc` and `o_dose` are valid PKNCA objects:
# 1) Sharing group variables in their formulas
# 2) Time units are the same within dose groups
# 3) Units are the same for subjects within the same concentration group

d_conc <- data.frame(
  subj = 1,
  analyte = rep(c("A", "B"), each = 2),
  concu = rep(c("ng/mL", "ug/mL"), each = 2),
  conc = c(0, 2, 0, 5),
  time = rep(0:1, 2),
  timeu = "h"
```

```

)
d_dose <- data.frame(
  subj = 1,
  dose = 100,
  doseu = "mg",
  time = 0,
  timeu = "h"
)
o_conc <- PKNCA::PKNCAconc(d_conc, conc ~ time | subj / analyte, concu = "concu")
o_dose <- PKNCA::PKNCAdose(d_dose, dose ~ time | subj, doseu = "doseu")
units_table <- PKNCA_build_units_table(o_conc, o_dose)

```

PKNCA_calculate_nca *Calculates results for PKNCA analysis.*

Description

Calculates results for PKNCA analysis.

Usage

```
PKNCA_calculate_nca(pknca_data, blq_rule = NULL)
```

Arguments

pknca_data	Data object created using PKNCA::PKNCAdata() function.
blq_rule	A list defining the Below Limit of Quantification (BLQ) imputation rule using PKNCA format. The list should either contain three elements named: first, middle, and last or two elements named before.tmax and after.tmax. Each element can be a numeric value (substituting the BLQ value), or a string such as "drop" (ignores the value) or "keep" (keeps the value as 0). Default is NULL, which does not specify any BLQ function to use for imputation. It is required if blq is defined in the intervals impute column of the pknca_data object, as the function will be applied to those intervals during the NCA calculation.

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.

Value

Results object with start and end times for each dose, from first dose and from most recent dose

Examples

```

example_data <- data.frame(
  STUDYID = rep("STUDY001", 6),
  PCSPEC = rep("Plasma", 6),
  ROUTE = rep("IV", 6),
  DOSETRT = rep("DrugA", 6),
  USUBJID = rep("SUBJ001", 6),
  ATPTREF = rep(1, 6),
  PARAM = rep("AnalyteA", 6),
  AVAL = c(0, 5, 10, 7, 3, 1),
  AVALU = rep("ng/mL", 6),
  DOSEA = rep(100, 6),
  DOSEU = rep("mg", 6),
  AFRLT = c(0, 1, 2, 3, 4, 6),
  ARRLT = c(0, 1, 2, 3, 4, 6),
  NFRLT = c(0, 1, 2, 3, 4, 6),
  ADOSEDUR = rep(0.5, 6),
  RRLTU = rep("hour", 6)
)

# Create a PKNCAdata object
pknca_data <- PKNCA_create_data_object(example_data)

# Perform NCA calculations
nca_results <- PKNCA_calculate_nca(pknca_data)

```

PKNCA_create_data_object

Creates a PKNCA::PKNCAdata object.

Description

Creates a PKNCA::PKNCAdata object.

Usage

```

PKNCA_create_data_object(
  adnca_data,
  mapping = NULL,
  applied_filters = NULL,
  time_duplicate_rows = NULL
)

```

Arguments

adnca_data Data frame containing raw or pre-processed ADNCA data.

mapping	Optional named list of column mappings (as produced by the Shiny mapping UI). When provided, the preprocessing pipeline is run internally and NCA exclusion flag columns are derived from mapping\$NCAwXRS (plus "DTYPE"). Metabolite names are taken from mapping\$Metabolites. Defaults to NULL (no preprocessing, no exclusion columns).
applied_filters	Optional list of filters to apply (see apply_filters()). Only used when mapping is provided. Defaults to NULL.
time_duplicate_rows	Optional integer vector of row indices (in the mapped dataset, before filtering) to mark as "TIME DUPLICATE" in the DTYPE column. When NULL (the default) and time duplicates are detected, an error of class "time_duplicate_error" is raised with the duplicate rows attached. Use this to forward user-resolved selections from the Shiny duplicate resolution modal.

Details

This function creates a standard PKNCAdata object from raw or pre-processed ADNCA data.

When mapping is provided, the function runs the full preprocessing pipeline internally: column mapping ([apply_mapping\(\)](#)), metabolite flagging ([create_metabfl\(\)](#)), class/length adjustment ([adjust_class_and_length\(\)](#)), duplicate annotation, optional filtering ([apply_filters\(\)](#)), and derives NCA exclusion flag columns from mapping\$NCAwXRS (plus "DTYPE").

When mapping is NULL (the default), the function expects already pre-processed ADNCA data with the standard column names in place.

The ADNCA data (after preprocessing, if applicable) must contain:

- STUDYID: Study identifier.
- PCSPEC: Matrix.
- ROUTE: Route of administration.
- DOSETRT: Drug identifier.
- USUBJID: Unique subject identifier.
- ATPTREF: (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.

Then it proceeds to:

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 ~ AFRLT | STUDYID + PCSPEC + DOSETRT + USUBJID / PARAM`.
4. Creating PKNCAdose object using `PKNCA::PKNCAdose()` with formula `DOSEA ~ AFRLT | STUDYID + DOSETRT + USUBJID`.
5. Creating PKNCAdata object using `PKNCA::PKNCAdata()`.
6. Updating units in PKNCAdata object so each analyte has its own unit.

Value

PKNCAdata object with concentration, doses, and units based on ADNCA data.

Examples

```
adnca_data <- data.frame(
  STUDYID = rep("STUDY001", 6),
  PCSPEC = rep("Plasma", 6),
  ROUTE = rep("IV", 6),
  DOSETRT = rep("DrugA", 6),
  USUBJID = rep("SUBJ001", 6),
  ATPTREF = rep(1, 6),
  PARAM = rep("AnalyteA", 6),
  AVAL = c(0, 5, 10, 7, 3, 1),
  AVALU = rep("ng/mL", 6),
  DOSEA = rep(100, 6),
  DOSEU = rep("mg", 6),
  AFRLT = c(0, 1, 2, 3, 4, 6),
  ARRLT = c(0, 1, 2, 3, 4, 6),
  NFRLT = c(0, 1, 2, 3, 4, 6),
  ADOSEDUR = rep(0.5, 6),
  RRLTU = rep("hour", 6)
)
PKNCA_create_data_object(adnca_data)
```

PKNCA_hl_rules_exclusion

Exclude NCA results based on user-defined rules over the half-life related parameters This function applies exclusion rules to the NCA results based on user-defined parameters.

Description

Exclude NCA results based on user-defined rules over the half-life related parameters This function applies exclusion rules to the NCA results based on user-defined parameters.

Usage

```
PKNCA_h1_rules_exclusion(res, rules)
```

Arguments

<code>res</code>	A PKNCAresults object containing the NCA results.
<code>rules</code>	A list of exclusion rules where each rule is a named vector.

Details

The function iterates over the rules and applies the exclusion criteria to the NCA results. For any parameter that is not `aucpext.obs` or `aucpext.pred` it applies a minimum threshold, and for `aucpext.obs` and `aucpext.pred` it applies a maximum threshold.

Value

A PKNCAresults object with the exclusions applied.

PKNCA_impute_method_start_c1

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

Description

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

Usage

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

Arguments

<code>conc</code>	Numeric vector of concentrations.
<code>time</code>	Numeric vector of times corresponding to the concentrations.
<code>start</code>	Numeric value indicating the start/dose time.
<code>end</code>	Numeric value indicating the end time.
<code>...</code>	Additional arguments (currently not used).
<code>options</code>	List of options (currently not used).

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](#).

Value

A data frame with imputed start concentration.

Examples

```
conc <- c(1, 2, 3, 4, 5)
time <- c(1, 2, 3, 4, 5)
start <- 0
end <- 4
PKNCA_impute_method_start_c1(conc, time, start, end)
```

PKNCA_impute_method_start_logslope

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

Description

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

Usage

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

Arguments

conc	Numeric vector of concentrations.
time	Numeric vector of times corresponding to the concentrations.
start	Numeric value indicating the start/dose time.
end	Numeric value indicating the end time.
...	Additional arguments (currently not used).
options	List of options (currently not used).

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](#).

Value

A data frame with imputed start concentration.

Examples

```
conc <- c(5, 4, 3, 2, 1)
time <- c(1, 2, 3, 4, 5)
start <- 0
end <- 4
PKNCA_impute_method_start_logslope(conc, time, start, end)
```

PKNCA_update_data_object

Create a PKNCAdata Object for NCA or Slope Analysis

Description

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

Usage

```
PKNCA_update_data_object(
  adnca_data,
  method,
  selected_analytes,
  selected_profile,
  selected_pcspec,
  start_impute = TRUE,
  hl_adj_rules = NULL,
  exclusion_list = NULL,
  keep_interval_cols = NULL,
  min_hl_points = 3,
  parameter_selections = NULL,
  int_parameters = NULL,
  blq_imputation_rule = NULL,
  custom_units_table = NULL
)
```

Arguments

adnca_data	A reactive PKNCAdata object
method	NCA calculation method selection
selected_analytes	User selected analytes
selected_profile	User selected dose numbers/profiles

selected_pcspec	User selected specimen
start_impute	Logical indicating whether to impute start concentration values. Also forwarded to update_main_intervals() when parameter_selections is provided.
hl_adj_rules	A data frame containing half-life adjustment rules. It must contain group columns and rule specification columns; TYPE: (Inclusion, Exclusion), RANGE: (start-end).
exclusion_list	List of exclusion reasons and row indices to apply to the concentration data. Each item in the list should have: • reason: character string with the exclusion reason (e.g., "Vomiting") • rows: integer vector of row indices to apply the exclusion to
keep_interval_cols	Optional character vector of additional columns to keep in the intervals data frame and when the NCA is run (pk.nca) also in the results
min_hl_points	Minimum number of points to use for half-life calculation. Must be ≥ 2 . Default is 3 (PKNCA default).
parameter_selections	Optional named list of selected PKNCA parameters by study type (forwarded to update_main_intervals()).
int_parameters	Optional data frame containing partial AUC ranges (forwarded to update_main_intervals()).
blq_imputation_rule	Optional list defining the BLQ imputation rule (forwarded to update_main_intervals()).
custom_units_table	Optional data frame with PPSTRESU overrides. When provided, applied via dplyr::rows_update() on the PKNCAdata units table.

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

Step 6: Indicate points excluded / selected manually for half-life

Step 7 (optional): Update intervals with parameter selections per study type and partial AUC ranges via [update_main_intervals\(\)](#).

Step 8 (optional): Apply custom units table for PPSTRESU overrides.

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

Value

A fully configured PKNCAdata object.

prepare_plot_data	<i>Prepare Data for PK Dose QC Plotting</i>
-------------------	---

Description

A helper function that validates, processes, and combines concentration and dose data. It creates the unified legend and faceting variables and calculates the factor levels for the plot scales.

Usage

```
prepare_plot_data(  
  data_conc,  
  data_dose,  
  shape_var,  
  colour_var,  
  grouping_vars,  
  labels_df,  
  tooltip_vars,  
  plot_conc_data,  
  plot_dose_data  
)
```

Arguments

data_conc	A data.frame of concentration data.
data_dose	An optional data.frame of dosing data.
shape_var	Character. The column name from data_conc for the legend.
colour_var	Character. The column name from data_dose for the legend.
grouping_vars	Character vector. Column names for faceting.
labels_df	A data.frame for label lookups.
tooltip_vars	Character vector of variables for the tooltip.
plot_conc_data	Logical flag derived from show_pk_samples and data_conc.
plot_dose_data	Logical flag derived from show_doses and data_dose.

Value

A list containing data (the processed tibble), shape_levels, and colour_levels.

read_pk	<i>Reads datasets from various file formats.</i>
---------	--

Description

Reads datasets from various file formats.

Usage

```
read_pk(path)
```

Arguments

path Character string with path to the dataset file.

Details

Currently supported file formats include:

- csv
- rds
- xlsx
- sas7bdat
- xpt
- parquet

Value

A data.frame object with loaded data.

Examples

```
path <- system.file("shiny/tests/testthat/dummy_simplified.csv", package = "aNCA")
df <- read_pk(path)
```

`read_settings`*Helper Logic to parse and structure settings YAML*

Description

Supports both the legacy flat format (top-level settings key) and the versioned format (top-level current key). For versioned files the most recent version is returned by default. Use `version` to select a specific version by index or comment. The full versioned object is attached as attribute "versioned" so callers can offer version selection.

Usage

```
read_settings(path, version = 1L)
```

Arguments

<code>path</code>	Character string with path to the settings YAML file.
<code>version</code>	Version selector for versioned settings files. Either an integer index (1 = most recent, 2 = second, etc.) or a character string matched against the version comment field. Defaults to 1L (most recent). Ignored for non-versioned (legacy) settings files.

Value

A list with parsed settings. For versioned files, the attribute "versioned" contains the full `read_versioned_settings()` result.

`read_versioned_settings`*Read a versioned settings YAML file*

Description

Parses a YAML file in the versioned format (with current and optionally previous keys). Returns a list with versions (a list of version entries, most recent first).

Usage

```
read_versioned_settings(path, obj = NULL)
```

Arguments

<code>path</code>	Character string with path to the YAML file.
<code>obj</code>	Optional pre-parsed YAML list. When supplied, <code>path</code> is ignored and no file I/O occurs.

Value

A list with element versions.

remove_pp_not_requested

Filter Out Parameters Not Requested in PKNCA Results (Pivot Version)

Description

This function removes parameters from the PKNCA results that were not requested by the user, using a pivoted approach.

Usage

```
remove_pp_not_requested(pknca_res)
```

Arguments

pknca_res A PKNCA results object containing at least \$data\$intervals and \$result.

Value

The PKNCA results object with non requested parameters removed from \$result.

rm_impute_obs_params *Apply Imputation for PKNCA Data*

Description

Applies imputation rules to a PKNCAdata object, imputing start values and then selectively removing imputation for parameters that are not dependent on AUC.

Usage

```
rm_impute_obs_params(data, metadata_nca_parameters = metadata_nca_parameters)
```

Arguments

data A PKNCAdata object.

metadata_nca_parameters

A data frame mapping PKNCA parameters (PKNCA) and information on their parameter dependencies ('Depends').

Value

A PKNCAdata object with imputation rules applied.

run_app*Run the Shiny app*

Description

Run the Shiny app

Usage

```
run_app(datapath = NULL, settings = NULL, settings_version = 1L, ...)
```

Arguments

datapath	Full path to a single .csv or .rds data file.
settings	Full path to a .yaml settings file.
settings_version	Version selector for versioned settings files. Either an integer index (1 = most recent, 2 = second, etc.) or a character string matched against the version comment field. Defaults to 1L (most recent). Ignored for non-versioned files.
...	Arguments passed to shiny::runApp()

Details

If a datapath is provided, the app will attempt to automatically load the specified dataset on startup. This is achieved by setting an internal option (`options(aNCA.datapath = datapath)`), which the app then reads. **This pre-loaded dataset can be overwritten; if a new file is uploaded using the widget within the app, it will replace the initial data for the current session.**

If datapath is NULL (default), the app will launch without pre-loading any data, and a file must be uploaded manually within the app.

If a settings path is provided, the app will load the settings file on startup and apply the saved configuration (analyte, method, units, etc.). The pre-loaded settings can be overwritten by uploading a new settings file.

For versioned settings files (containing current and previous entries), the most recent version is loaded by default. Use settings_version to select a different version by index or comment:

```
run_app(settings = "settings.yaml", settings_version = 2)
run_app(settings = "settings.yaml", settings_version = "NCA draft")
```

Value

No return value, called for side effects to launch the Shiny application.

Examples

```
## Not run:
# Launch the app without pre-loaded data or settings:
run_app()

# Pre-load a settings file (use the full path to your YAML file):
run_app(settings = system.file("www/templates/clinical_template.yaml", package = "aNCA"))

# Pre-load both data and settings:
run_app(
  datapath = "/path/to/adpc.csv",
  settings = "/path/to/settings.yaml"
)

# Load a specific version from a versioned settings file:
run_app(settings = "/path/to/settings.yaml", settings_version = 2)
run_app(settings = "/path/to/settings.yaml", settings_version = "NCA draft")

## End(Not run)
```

settings_version_summary

Build a summary table of version entries

Description

Creates a data.frame with one row per version, suitable for display in a modal.

Usage

```
settings_version_summary(versions)
```

Arguments

versions List of version entries.

Value

A data.frame with columns: index, comment, datetime, dataset, anca_version, tab.

simplify_unit	<i>Simplify compound unit expressions</i>
---------------	---

Description

This function takes a units object or a character string representing a unit expression and returns a simplified units using the units package simplifications.

Usage

```
simplify_unit(x, as_character = FALSE)
```

Arguments

x	A units object, character string, or vector of either to be simplified.
as_character	Logical. TRUE returns the result as a character, FALSE (default) as a unit object.

Value

A simplified units object, or a list of units objects if input is a vector.

Examples

```
# Using a units object
u <- units::set_units(1, "L*g/mg", mode = "standard")
simplify_unit(u)

# Using a character string
simplify_unit("(mg*L)/(mL)")
```

t_pkct01	<i>Summary Concentration Table (pkct01)</i>
----------	---

Description

Summarizes PK concentration data by treatment/dose group and nominal timepoint. Returns one data frame per analyte/specimen combination containing descriptive statistics across subjects at each scheduled timepoint.

Usage

```

t_pkct01(
  data,
  list_vars = c("PARAM", "PCSPEC"),
  strat_var = c("TRT01A", "ATPTREF", "NFRLT"),
  value_var = "AVAL",
  blq_var = "AVALC",
  time_var = "NFRLT",
  time_filter = NULL,
  col_group_var = NULL,
  stats = NULL
)

t_pkct01_dose(data, strat_var = c("DOSEA", "ATPTREF", "NFRLT"), ...)

t_pkct01_tad(data, strat_var = c("TRT01A", "ATPTREF", "NRRLT"), ...)

t_pkct01_dose_tad(data, strat_var = c("DOSEA", "ATPTREF", "NRRLT"), ...)

```

Arguments

data	A CDISC ADNCA data frame (from <code>export_cdisc()</code> \$adnca).
list_vars	Character vector of columns used to split the output into separate tables. Default: <code>c("PARAM", "PCSPEC")</code> .
strat_var	One or more columns whose combination defines the table rows (stratification). Default: <code>c("TRT01A", "ATPTREF", "NFRLT")</code> – treatment arm, visit reference, and nominal timepoint. Add or remove columns to change how the rows are grouped. Any variable that is also a <code>list_vars</code> (table-split) column is dropped from the rows, since it is constant within each split.
value_var	Column containing the numeric analysis value. Default: "AVAL".
blq_var	Column containing the character analysis value used to detect BLQ records. Default: "AVALC". Records where this column equals "BLQ" are counted separately and excluded from numeric summaries. When <code>blq_var</code> is absent (as in <code>export_cdisc()</code> \$adnca, which does not include AVALC), BLQ is detected via <code>value_var == 0</code> , consistent with the package convention for post-imputation BLQ encoding.
time_var	Column that the <code>time_filter</code> applies to (the nominal timepoint column). Default: "NFRLT". Row grouping is controlled entirely by <code>strat_var</code> ; this argument only names the column that <code>time_filter</code> subsets.
time_filter	Optional vector of <code>time_var</code> values to keep. NULL (default) keeps every time-point.
col_group_var	Optional subject-level column (e.g. "SEX", "RACE") whose values become side-by-side comparison column groups: the full statistic block is repeated once per level, nested under a group header. NULL (default) produces the standard flat table. Must differ from <code>strat_var</code> and the <code>list_vars</code> .

stats	Optional character vector of statistics to display, chosen from c("n", "n_blq", "Mean", "SD", "CV_pct", "Median", "GeoMean", "GeoCV_pct", "Min", "Max"). NULL (default) shows all of them.
...	Additional arguments forwarded to <code>t_pkct01()</code> .

Details

BLQ values are excluded from all numeric statistics and counted in n_blq. When blq_var is present, BLQ is identified as df[[blq_var]] == "BLQ". When blq_var is absent, value_var == 0 is used as the fallback BLQ indicator. GeoMean is computed on positive value_var values only.

Value

A named list of data frames, one per unique combination of list_vars. Each data frame has one column per strat_var followed by the statistics: n, n_blq, Mean, SD, CV_pct, Median, GeoMean, GeoCV_pct, Min, Max. When col_group_var is set, the statistic columns are prefixed per group level and a col_groups attribute drives the rendered two-level header.

Functions

- t_pkct01_dose(): Stratify by dose instead of treatment arm (first dose).
- t_pkct01_tad(): Summarize using time after dose (TAD) nominal time.
- t_pkct01_dose_tad(): Stratify by dose using TAD nominal time.

Examples

```
## Not run:
adnca <- export_cdisc(res_nca)$adnca
tables <- t_pkct01(adnca)
tables[[1]]

## End(Not run)
```

t_pkpt03_col	Summary PK Parameters Table – statistics in columns (pkpt03)
--------------	--

Description

Summarizes pharmacokinetic parameters from ADPP data. Returns one data frame per analyte (PPCAT) combination with PK parameters as rows and descriptive statistics as columns.

Usage

```
t_pkpt03_col(
  data,
  list_vars = c("PPCAT"),
  strat_var = c("TRT01A", "PARAM"),
  value_var = "AVAL",
  param_filter = NULL,
  col_group_var = NULL,
  stats = NULL
)

t_pkpt03_MP_col(data, ...)
```

Arguments

data	A CDISC ADPP data frame (from <code>export_cdisc()</code> \$adpp).
list_vars	Character vector of columns used to split output into separate tables. Default: <code>c("PPCAT")</code> . AVISIT is a conditional ADPP column that is typically absent from <code>export_cdisc()</code> \$adpp; it is silently skipped when not present so there is no need to remove it manually, but adding it only helps when your ADPP actually contains visit information.
strat_var	One or more columns whose combination defines the table rows (stratification). Default: <code>c("TRT01A", "PARAM")</code> – statistics are separated by treatment arm and parameter. Add e.g. "PCSPEC" to also split by specimen. Any variable that is also a <code>list_vars</code> (table-split) column is dropped from the rows, since it is constant within each split.
value_var	Column containing the numeric analysis value. Default: "AVAL".
param_filter	Optional character vector of PARAM values to keep. NULL (default) keeps every parameter.
col_group_var	Optional subject-level column (e.g. "SEX", "RACE") whose values become side-by-side comparison column groups: the full statistic block is repeated once per level, nested under a group header. NULL (default) produces the standard flat table. Must differ from <code>strat_var</code> and the <code>list_vars</code> .
stats	Optional character vector of statistics to display, chosen from <code>c("n", "Mean", "SD", "CV_pct", "GeoMean", "GeoCV_pct", "Median", "Min", "Max")</code> . NULL (default) shows all of them. Names not produced by this table are ignored.
...	Additional arguments forwarded to <code>t_pkpt03_col()</code> .

Value

A named list of data frames, one per combination of `list_vars`. Each data frame has one column per `strat_var` followed by the statistics: n, Mean, SD, CV_pct, GeoMean, GeoCV_pct, Median, Min, Max. When `col_group_var` is set, the statistic columns are prefixed per group level and a `col_groups` attribute drives the rendered two-level header.

Functions

- t_pkpt03_MP_col(): Summary of metabolite-to-parent ratios (stats in columns). Filters to metabolite rows using METABFL (preferred) or, when absent from ADPP, falls back to rows where PPCAT or PARAM contains "metab" (case-insensitive). METABFL is present in ADPP only when it was included as a grouping variable in the NCA run.

Examples

```
## Not run:
adpp <- export_cdisc(res_nca)$adpp
tables <- t_pkpt03_col(adpp)
tables[[1]]
# Separate statistics by specimen too:
t_pkpt03_col(adpp, strat_var = c("TRT01A", "PARAM", "PCSPEC"))[[1]]
# Compare sexes side by side:
t_pkpt03_col(adpp, col_group_var = "SEX")[[1]]

## End(Not run)
```

t_pkpt07_norm	Mean Dose-Normalized PK Parameters Table (pkpt07)
---------------	---

Description

Filters ADPP to dose-normalized parameters and summarizes them with the same column layout as [t_pkpt03_col\(\)](#). These parameters must have been computed during the NCA run – they are not derived on the fly.

Usage

```
t_pkpt07_norm(
  data,
  paramcd_var = "PARAMCD",
  paramcd_filter = c("CMAXD", "AUCLSTD", "AUCIFOD", "AUCTLSTD"),
  list_vars = c("PPCAT"),
  strat_var = c("TRT01A", "PARAM"),
  value_var = "AVAL",
  col_group_var = NULL,
  stats = NULL
)
```

Arguments

data	A CDISC ADPP data frame (from export_cdisc())\$adpp).
paramcd_var	Column containing parameter codes used to detect dose-normalized parameters. Default: "PARAMCD".

paramcd_filter	Character vector of CDISC dose-normalized PARAMCDs to keep. Defaults to the standard codes used in this package: <code>c("CMAXD", "AUCLSTD", "AUCIFOD", "AUCLSTD")</code> . Pass <code>NULL</code> to fall back to the regex <code>grep1("[A-Z0-9]D\$", PARAMCD)</code> pattern, which keeps any code whose last two characters are an uppercase letter/digit followed by D.
list_vars	Character vector of columns used to split output into separate tables. Default: <code>c("PPCAT")</code> . AVISIT is a conditional ADPP column that is typically absent from <code>export_cdisc()</code> \$adpp; it is silently skipped when not present so there is no need to remove it manually, but adding it only helps when your ADPP actually contains visit information.
strat_var	One or more columns whose combination defines the table rows (stratification). Default: <code>c("TRT01A", "PARAM")</code> – statistics are separated by treatment arm and parameter. Add e.g. "PCSPEC" to also split by specimen. Any variable that is also a <code>list_vars</code> (table-split) column is dropped from the rows, since it is constant within each split.
value_var	Column containing the numeric analysis value. Default: "AVAL".
col_group_var	Optional subject-level column (e.g. "SEX", "RACE") whose values become side-by-side comparison column groups: the full statistic block is repeated once per level, nested under a group header. <code>NULL</code> (default) produces the standard flat table. Must differ from <code>strat_var</code> and the <code>list_vars</code> .
stats	Optional character vector of statistics to display, chosen from <code>c("n", "Mean", "SD", "CV_pct", "GeoMean", "GeoCV_pct", "Median", "Min", "Max")</code> . <code>NULL</code> (default) shows all of them. Names not produced by this table are ignored.

Value

Named list of data frames (same format as `t_pkpt03_col()`).

Examples

```
## Not run:
adpp <- export_cdisc(res_nca)$adpp
tables <- t_pkpt07_norm(adpp)
# Include a custom dose-normalized code:
tables <- t_pkpt07_norm(adpp, paramcd_filter = c("CMAXD", "AUCLSTD", "MYPARAMD"))

## End(Not run)
```

t_pkpt08_uri	<i>Mean Urine Amount and Percent Recovered Table (pkpt08)</i>
--------------	---

Description

Filters ADPP to urine specimen records and summarizes cumulative amount excreted (Ae) and percentage of dose recovered (Fe%) with descriptive statistics in columns. Per the TLG catalog specification for pkpt08, the summary includes n, Mean, SD, CV%, Median, Min, Max – without geometric mean or geometric CV% (those are omitted because urine recovery parameters are not log-normally distributed by convention).

Usage

```
t_pkpt08_uri(
  data,
  urine_specs = c("URINE"),
  list_vars = c("PPCAT"),
  strat_var = c("TRT01A", "PARAM"),
  value_var = "AVAL",
  param_filter = NULL,
  col_group_var = NULL,
  stats = NULL
)
```

Arguments

data	A CDISC ADPP data frame. Urine records are identified by PPSPEC %in% urine_specs.
urine_specs	Character vector of specimen types considered urine, matched case-insensitively. Default: c("URINE").
list_vars	Character vector of columns used to split output into separate tables. Default: c("PPCAT"). AVISIT is a conditional ADPP column that is typically absent from export_cdisc()\$adpp; it is silently skipped when not present so there is no need to remove it manually, but adding it only helps when your ADPP actually contains visit information.
strat_var	One or more columns whose combination defines the table rows (stratification). Default: c("TRT01A", "PARAM") – statistics are separated by treatment arm and parameter. Add e.g. "PCSPEC" to also split by specimen. Any variable that is also a list_vars (table-split) column is dropped from the rows, since it is constant within each split.
value_var	Column containing the numeric analysis value. Default: "AVAL".
param_filter	Optional character vector of PARAM values to keep. NULL (default) keeps every parameter.
col_group_var	Optional subject-level column (e.g. "SEX", "RACE") whose values become side-by-side comparison column groups: the full statistic block is repeated once per level, nested under a group header. NULL (default) produces the standard flat table. Must differ from strat_var and the list_vars.
stats	Optional character vector of statistics to display, chosen from c("n", "Mean", "SD", "CV_pct", "GeoMean", "GeoCV_pct", "Median", "Min", "Max"). NULL (default) shows all of them. Names not produced by this table are ignored.

Value

Named list of data frames with one column per strat_var followed by n, Mean, SD, CV_pct, Median, Min, Max. Use `t_pkpt03_col()` instead if geometric mean statistics are needed.

Examples

```
## Not run:
adpp <- export_cdisc(res_nca)$adpp
tables <- t_pkpt08_uri(adpp)

## End(Not run)
```

t_pkpt11_gmr	<i>GMR Table with Confidence Intervals (pkpt11)</i>
--------------	---

Description

Computes geometric mean ratios (GMR) with 90% confidence intervals for selected PK parameters, comparing each treatment arm to a reference arm.

Usage

```
t_pkpt11_gmr(
  data,
  ref_arm = NULL,
  ci_level = 0.9,
  list_vars = c("PPCAT"),
  strat_var = "TRT01A",
  param_var = "PARAM",
  value_var = "AVAL"
)
```

Arguments

data	A CDISC ADPP data frame.
ref_arm	Character string identifying the reference treatment arm in strat_var. If NULL (default), the first arm in sorted order is used.
ci_level	Confidence level for the geometric mean ratio CI. Default: 0.90.
list_vars	Character vector of columns used to split output into separate tables. Default: c("PPCAT"). AVISIT is a conditional ADPP column that is typically absent from export_cdisc()\$adpp; it is silently skipped when not present so there is no need to remove it manually, but adding it only helps when your ADPP actually contains visit information.
strat_var	Single treatment-arm column that defines the comparison axis (each arm is compared against ref_arm). Default: "TRT01A". Unlike the summary tables, this must be a single column.

param_var	Column containing parameter names shown as rows. Default: "PARAM".
value_var	Column containing the numeric analysis value. Default: "AVAL".

Details

The confidence interval is computed on the log scale using a two-sample t-test approach: $\exp(\log_ratio \pm t * SE)$ where SE is derived from the pooled within-group standard deviations on the log scale.

Value

Named list of data frames, one per combination of list_vars. Each data frame has columns: strat_var, param_var, n_ref, n_trt, GMR, CI_lower, CI_upper.

Examples

```
## Not run:
adpp <- export_cdisc(res_nca)$adpp
tables <- t_pkpt11_gmr(adpp, ref_arm = "Placebo")

## End(Not run)
```

translate_terms	<i>Translate Terms from One Nomenclature to Another</i>
-----------------	---

Description

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

Usage

```
translate_terms(
  input_terms,
  mapping_col = "PKNCA",
  target_col = "PPTTESTCD",
  metadata = metadata_nca_parameters
)
```

Arguments

input_terms	A character vector of terms to be translated.
mapping_col	Character indicating the column name in the metadata file of the input terms. Default is "PKNCA".
target_col	Character indicating the column name in the metadata file of the target terms. Default is "PPTTESTCD".
metadata	Dataset used to do the mapping that contains the mapping and target columns.

Value

A character vector of translated terms. Input terms not available in mapping_col will be returned with the same value.

Examples

```
input_terms <- c("adj.r.squared", "ae", "nonexistent_term")
translate_terms(input_terms)
```

update_main_intervals *Update an intervals data frame with user-selected parameters by study type*

Description

Update an intervals data frame with user-selected parameters by study type

Usage

```
update_main_intervals(
  data,
  parameter_selections = NULL,
  int_parameters = NULL,
  start_impute = TRUE,
  blq_imputation_rule = NULL
)
```

Arguments

data	A PKNCAdata object containing intervals and dosing data.
parameter_selections	A named list of selected PKNCA parameters by study type.
int_parameters	A data frame containing partial AUC ranges.
start_impute	Logical indicating whether to impute start values for parameters. Named start_impute (not impute) to avoid colliding with the per-interval impute column referenced under dplyr data-masking below.
blq_imputation_rule	A list defining the Below Limit of Quantification (BLQ) imputation rule using PKNCA format. The list should either contain three elements named: first, middle, and last or two elements named before.tmax and after.tmax. Each element can be a numeric value (substituting the BLQ value), or a string such as "drop" (ignores the value) or "keep" (keeps the value as 0). Default is NULL, which does not specify any BLQ imputation in any interval.

Value

An updated PKNCAdata object with parameter intervals based on user selections.

update_pknca_with_rules

Apply Slope Rules to Update Data

Description

Iterates over the given rules and updates the PKNCA object setting inclusion/exclusion flags.

Usage

```
update_pknca_with_rules(data, slopes)
```

Arguments

data	PKNCA data object
slopes	Data frame of slope rules (TYPE, RANGE, REASON, group columns)

Value

Modified data object with updated flags

validate_cdisc_types *Validate CDISC datasets against metadata-declared data types*

Description

Checks each column of the supplied CDISC datasets against the expected storage class and maximum length declared in metadata_nca_variables. This is intentionally not a full CDISC conformance engine: it covers only what the metadata can assert (class and length), and never fails on metadata Type values it cannot interpret.

Usage

```
validate_cdisc_types(
  cdisc_data,
  metadata = metadata_nca_variables,
  check_length = TRUE,
  grouping_vars = character(0)
)
```

Arguments

<code>cdisc_data</code>	Named list of CDISC data frames keyed <code>pp</code> , <code>adpp</code> , and/or <code>adnca</code> (as returned by <code>export_cdisc()</code>). Entries with other names or that are not data frames are ignored.
<code>metadata</code>	Metadata data frame with <code>Dataset</code> , <code>Variable</code> , <code>Type</code> , and <code>Length</code> columns. Defaults to <code>metadata_nca_variables</code> .
<code>check_length</code>	Logical. Check character values against the declared <code>Length</code> ? Defaults to <code>TRUE</code> .
<code>grouping_vars</code>	Character vector of explicitly selected grouping variables. Undeclared variables in this vector are allowed in ADPP and ADNCA when they contain supported atomic vectors, with a warning that metadata type/length checks are unavailable. PP remains strict. Known metadata rules always apply.

Details

Indexed CDISC variable families are resolved to their metadata template: a data column falls back from an exact name match to an index-agnostic match, so `NCA1XRS/NCA2XRS` match the `NCAwXRS` template and `CRIT2/CRIT3FL` match the `CRIT1/CRIT1FL` family. An exact metadata entry always takes precedence over the template.

Value

A tidy findings data frame with columns `Dataset`, `Variable`, `Check`, `Severity`, `Expected`, `Observed`, `N_Affected`, and `Message`. Zero rows means the data conforms.

Examples

```
cdisc_data <- list(
  adnca = data.frame(STUDYID = "S1", AVAL = 1.2, stringsAsFactors = FALSE)
)
validate_cdisc_types(cdisc_data)
```

```
validate_export_outputs
```

Validate exported outputs before saving

Description

Checks every output that the export/Save flow would write, so a save can be refused before any file is created. It combines two levels of checking:

Usage

```
validate_export_outputs(
  output,
  obj_names = NULL,
  metadata = metadata_nca_variables,
  grouping_vars = character(0)
)
```

Arguments

output	The export list as assembled for the Save flow: a named, possibly nested list of outputs (plots, tables, and *_code strings), with the CDISC datasets under a CDISC element keyed pp/adpp/adnca.
obj_names	Optional character vector of leaf names that will be written. When NULL (the default) every leaf is validated.
metadata	Metadata data frame passed to validate_cdisc_types() for the CDISC value-level checks. Defaults to metadata_nca_variables.
grouping_vars	Selected custom grouping variables allowed in ADPP/ADNCA, passed to validate_cdisc_types() . Known fields and PP remain strictly checked.

Details

- **Object class** for all outputs: plot nodes must be a ggplot or plotly, table nodes must be a data frame, and *_code nodes must be a length-1 character string. Anything the export writer cannot serialise is reported as an error.
- **Value level** for the CDISC datasets: the pp/adpp/adnca entries under CDISC are additionally checked against the metadata-declared data types and lengths via [validate_cdisc_types\(\)](#).

The walk mirrors the export writer: plain lists are recursed into, and when obj_names is supplied only those named leaves (the ones that will actually be written) are validated.

Value

A tidy findings data frame with columns Output, Variable, Check, Severity, Expected, Observed, and Message. Variable is NA for object-class findings and holds the column name for CDISC value-level findings. Zero rows means every output is safe to export.

Examples

```
output <- list(
  nca_results = list(nca_pkparam = data.frame(AVAL = 1.2)),
  CDISC = list(adnca = data.frame(STUDYID = "S1", AVAL = 1.2))
)
validate_export_outputs(output)
```

validate_pk	<i>Validates data table with raw pk data.</i>
-------------	---

Description

Validates data table with raw pk data.

Usage

```
validate_pk(pk_data)
```

Arguments

pk_data Object to check.

Details

Performs the following checks:

- If provided variable is of class `data.frame`.
- If number of rows in provided data frame is greater than 0.

Throws an error if any of the checks does not pass.

Value

Original, unchanged object. If any of the checks do not pass, throws an error.

write_versioned_settings

Write a versioned settings YAML file

Description

Writes a list of version entries to a YAML file in the versioned format. The first entry becomes current, the rest go under previous.

Usage

```
write_versioned_settings(versions, path)
```

Arguments

versions A list of version entries (as returned by `create_settings_version()`).

path Character string with the output file path.

Value

Invisibly returns path.

Index

* datasets

- adnca_example, [13](#)
- dm_example, [30](#)
- ex_example, [32](#)
- metadata_nca_parameters, [63](#)
- metadata_nca_variables, [64](#)
- pc_example, [73](#)
- .compress_range, [5](#)
- .create_filter_expr, [5](#)
- .eval_range, [6](#)
- .handle_tooltips, [6](#)
- add_exclusion_reasons, [7](#)
- add_label_attribute, [7](#)
- add_pptx_sl_plot, [8](#)
- add_pptx_sl_plottable, [9](#)
- add_pptx_sl_table, [9](#)
- add_qmd_plot, [10](#)
- add_qmd_sl_plot, [10](#)
- add_qmd_sl_plottabletable, [11](#)
- add_qmd_table, [11](#)
- add_settings_version, [12](#)
- adjust_class_and_length, [12](#)
- adjust_class_and_length(), [89](#)
- adnca_example, [13](#), [30–33](#), [73](#), [74](#)
- apply_filters, [14](#)
- apply_filters(), [89](#)
- apply_labels, [15](#)
- apply_mapping, [16](#)
- apply_mapping(), [89](#)
- apply_unit_defaults, [17](#)
- calculate_ratio_app, [18](#)
- calculate_ratios, [19](#)
- calculate_summary_stats, [20](#)
- calculate_table_ratios, [21](#)
- cdisc_validation_blocks_save, [21](#)
- changed_units, [22](#)
- convert_volume_units, [22](#)
- create_metabfl, [24](#)

- create_metabfl(), [89](#)
- create_pptx_dose_slides, [24](#)
- create_qmd_doc, [25](#)
- create_qmd_dose_slides, [26](#)
- create_settings_version, [26](#)
- create_settings_version(), [12](#), [114](#)
- create_start_impute, [27](#)
- delete_settings_version, [27](#)
- detect_study_types, [28](#)
- dm_example, [30](#), [33](#), [74](#)
- dose_profile_duplicates, [31](#)
- dplyr::rows_update(), [94](#)
- ex_example, [30](#), [32](#)
- exploration_individualplot, [33](#)
- exploration_meanplot, [34](#)
- export_cdisc, [36](#)
- export_cdisc(), [112](#)
- export_validation_blocks_save, [37](#)
- filter_breaks, [37](#)
- flexible_violinboxplot, [38](#)
- format_pkncaconc_data, [39](#)
- format_pkncadata_intervals, [41](#)
- format_pkncadose_data, [42](#)
- format_unit_string, [43](#)
- g_lineplot, [43](#)
- g_pkcg01_lin, [45](#)
- g_pkcg01_log, [46](#)
- g_pkcg01_sbs, [46](#)
- g_pkcg02_lin, [47](#)
- g_pkcg02_log, [47](#)
- g_pkcg02_sbs, [48](#)
- g_pkcg03_lin, [48](#)
- g_pkcg03_log, [49](#)
- g_pkcg03_sbs, [49](#)
- generate_pre_specs, [50](#)
- generate_tooltip_text, [50](#)

- get_conversion_factor, 51
- get_half-life_plots, 52
- get_label, 52
- get_settings_code, 53
- interval_add_impute, 54
- interval_remove_impute, 55
- keep_blk_timepoints, 56
- l_pkcl01, 57
- l_pkcl01(), 60, 61
- l_pkcl01_tad, 59
- l_pkcl02_uri, 60
- l_pkpl01, 61
- l_pkpl01(), 61, 62
- l_pkpl01_mp (l_pkpl01), 61
- l_pkpl04_mp, 62
- metadata_nca_parameters, 63
- metadata_nca_variables, 64
- multiple_matrix_ratios, 65
- p_pkcg03_lin_dose, 66
- p_pkcg03_log_dose, 67
- p_pkcg03_sbs_dose, 67
- p_pkpg01_cum, 68
- p_pkpg01_cum(), 69
- p_pkpg01_per (p_pkpg01_cum), 68
- p_pkpg02_doseprop, 69
- p_pkpg03_boxp, 71
- p_pkpg03_boxp(), 71, 72
- p_pkpg04_boxp (p_pkpg03_boxp), 71
- p_pkpg06_mp, 72
- parse_annotation, 73
- pc_example, 30, 73
- pivot_wider_pknca_results, 74
- pk.calc.ermx, 75
- pk.calc.ertlst, 76
- pk.calc.ertmax, 76
- pk.calc.volpk, 77
- pk_dose_qc_plot, 78
- pkcg01, 80
- pkcg02, 82
- pkcg03, 84
- PKNCA_build_units_table, 86
- PKNCA_calculate_nca, 87
- PKNCA_create_data_object, 88
- PKNCA_hl_rules_exclusion, 90
- PKNCA_impute_method_start_c1, 91
- PKNCA_impute_method_start_logslope, 92
- PKNCA_update_data_object, 93
- prepare_plot_data, 95
- read_pk, 96
- read_settings, 97
- read_versioned_settings, 97
- read_versioned_settings(), 97
- remove_pp_not_requested, 98
- rm_impute_obs_params, 98
- run_app, 99
- settings_version_summary, 100
- simplify_unit, 101
- t_pkct01, 101
- t_pkct01(), 103
- t_pkct01_dose (t_pkct01), 101
- t_pkct01_dose_tad (t_pkct01), 101
- t_pkct01_tad (t_pkct01), 101
- t_pkpt03_col, 103
- t_pkpt03_col(), 104–106, 108
- t_pkpt03_MP_col (t_pkpt03_col), 103
- t_pkpt03_MP_col(), 62, 72
- t_pkpt07_norm, 105
- t_pkpt08_uri, 106
- t_pkpt11_gmr, 108
- translate_terms, 109
- update_main_intervals, 110
- update_main_intervals(), 94
- update_pknca_with_rules, 111
- validate_cdisc_types, 111
- validate_cdisc_types(), 22, 113
- validate_export_outputs, 112
- validate_export_outputs(), 37
- validate_pk, 113
- write_versioned_settings, 114