

# Package ‘PKbioanalysis’

February 17, 2026

**Type** Package

**Title** Pharmacokinetic Bioanalysis Experiments Design and Exploration

**Version** 0.5.0

**Maintainer** Omar Elashkar <omar.i.elashkar@gmail.com>

**Description** Automate pharmacokinetic/pharmacodynamic bioanalytical procedures based on best practices and regulatory recommendations.

The package impose regulatory constrains and sanity checking for common bioanalytical procedures.

Additionally, 'PKbioanalysis' provides a relational infrastructure for plate management and injection sequence.

**License** AGPL (>= 3)

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Depends** R (>= 4.3.0)

**Imports** dplyr, tidyselect, stringr (>= 1.5.1), ggplot2, ggforce (>= 0.4.1), tidyr, glue (>= 1.6.2), checkmate, shiny, DBI, duckdb (>= 1.0.0), bslib, bsicons, cli, DiagrammeR, DT, shinyWidgets, shinyjs, units, stats, shinyalert, htmltools, rlang, grDevices, utils, yaml, rhandsontable, methods, reticulate (>= 1.44.1), RTMB, janitor, tibble, xml2, RaMS, data.tree, nloptr, forcats, ggiraph (>= 0.9.3), gt, plotly, pracma, reactable, gtools, ellmer, htmlwidgets, jsonlite, nlme, scales, shinychat, uuid, sortable, writexl

**Suggests** rxode2, knitr, patchwork, rmarkdown, testthat (>= 3.0.0)

**SystemRequirements** python (>= 3.10)

**URL** <https://omarashkar.github.io/PKbioanalysis/>

**BugReports** <https://github.com/OmarAshkar/PKbioanalysis/issues>

**Config/testthat/edition** 3

**Collate** 'PKbioanalysis-package.R' 'calc.R' 'chrom\_anomaly\_plots.R' 'chrom\_app.R' 'chrom\_parsers.R' 'chrom\_utils.R' 'generics.R' 'class.R' 'chromatogram.R' 'config.R' 'dil\_map.R'

'estim\_residuals.R' 'genAI.R' 'gen\_studydesign.R'  
 'generate\_test\_set.R' 'injec\_list.R' 'linearitycheck.R'  
 'method\_file.R' 'peak\_integrate.R' 'plate.R' 'plate\_expr.R'  
 'plates\_class.R' 'process\_chroms.R' 'quant\_app.R'  
 'quant\_misc.R' 'quant\_object.R' 'quant\_parsers.R'  
 'samples\_profiles.R' 'study\_app.R' 'suitability\_utils.R'  
 'utils.R' 'writers.R' 'zzz.R'

**NeedsCompilation** no

**Author** Omar Elashkar [aut, cre] (ORCID:

<https://orcid.org/0000-0002-5505-778X>>)

**Repository** CRAN

**Date/Publication** 2026-02-17 21:40:07 UTC

## Contents

add_blank . . . . .	3
add_cs_curve . . . . .	4
add_DB . . . . .	5
add_DQC . . . . .	5
add_QC . . . . .	6
add_samples . . . . .	7
add_samples_db . . . . .	8
add_samples_db2 . . . . .	8
add_suitability . . . . .	9
area_report.PeakRes . . . . .	10
calc_var_summary . . . . .	10
check_chrom_cmpds . . . . .	11
chrom_app . . . . .	12
combine_injec_lists . . . . .	12
combine_plates . . . . .	13
config_suitability . . . . .	13
create_new_study . . . . .	14
cv . . . . .	14
download_sample_list . . . . .	15
estim_dil_limit . . . . .	15
estim_lloq . . . . .	16
export_integration . . . . .	16
export_pk_profiles . . . . .	17
export_run . . . . .	17
extract_peak_bounds . . . . .	18
fill_scheme . . . . .	18
filter_chrom . . . . .	19
fit_var . . . . .	20
formatted_print . . . . .	20
generate_96 . . . . .	21
get_compound_ID . . . . .	22

get_sample_ID . . . . .	22
get_sample_names . . . . .	23
has_default_bounds . . . . .	23
install_py_dep . . . . .	24
integrate . . . . .	24
is_integrated . . . . .	25
is_smoothed . . . . .	25
length,MultiPlate-method . . . . .	26
make_calibration_study . . . . .	26
make_metabolic_study . . . . .	27
nca_table . . . . .	28
pkmerge . . . . .	28
plate_metadata . . . . .	29
plate_tree . . . . .	29
plot.PlateObj . . . . .	30
plot_chrom . . . . .	31
plot_peak_areas.PeakRes . . . . .	32
plot_RT.ChromRes . . . . .	33
plot_RT.PeakRes . . . . .	33
plot_var_pattern . . . . .	34
precision_per_vial . . . . .	34
prefilter_precision_data . . . . .	35
quant_app . . . . .	36
read_chrom . . . . .	36
read_experiment_results . . . . .	37
register_plate . . . . .	37
response_to_conc . . . . .	38
reverse_predict . . . . .	38
run_summary . . . . .	39
smooth_chrom . . . . .	39
study_app . . . . .	40
update_RT . . . . .	40
write_injec_seq . . . . .	41
[[,MultiPlate-method . . . . .	42

## Index 43

---

add_blank	<i>Add blank to the plate Can be either double blank (DB), CS0IS+ or CS+IS0</i>
-----------	---

---

### Description

Add blank to the plate Can be either double blank (DB), CS0IS+ or CS+IS0

### Usage

```
add_blank(plate, IS = TRUE, analyte = FALSE, analytical = FALSE, group = NA)
```

**Arguments**

plate	PlateObj object
IS	logical. If TRUE, add IS to the well.
analyte	logical. If TRUE, add analyte to the well.
analytical	logical. If FALSE, the blank is analytical, if TRUE it is bioanalytical.
group	A string for bioanalytical group.

**Value**

PlateObj

---

add_cs_curve	<i>Add calibration curve to the plate</i>
--------------	---

---

**Description**

Add calibration curve to the plate

**Usage**

```
add_cs_curve(plate, plate_std, rep = 1, group = NA)
```

**Arguments**

plate	PlateObj
plate_std	character
rep	numeric. Number of technical replicates. Default is 1.
group	A string for bioanalytical group.

**Value**

PlateObj

**Examples**

```
plate <- generate_96() |>
  add_cs_curve(c(1, 3, 5, 10, 50, 100, 200))
plot(plate)
```

---

add_DB	<i>Add double blank (DB) to a plate</i>
--------	---

---

**Description**

Add double blank (DB) to a plate

**Usage**

```
add_DB(plate, analytical = FALSE, group = NA)
```

**Arguments**

plate	PlateObj object
analytical	logical. If TRUE, the blank is bioanalytical, if FALSE it is analytical.
group	A string for bioanalytical group.

**Value**

PlateObj

**Examples**

```
plate <- generate_96() |>  
add_DB()
```

---

add_DQC	<i>Add dilution quality control (DQC) to the plate</i>
---------	--

---

**Description**

Add dilution quality control (DQC) to the plate

**Usage**

```
add_DQC(plate, conc, fac, rep = 5, group = NA)
```

**Arguments**

plate	PlateObj object
conc	numeric. Concentration of the DQC well.
fac	numeric. Factor of the DQC well.
rep	numeric. Number of replicates. Default is 5.
group	A string for bioanalytical group. The current implementation does not check ULOQ or LLOQ boundaries.

---

`add_QC`*Add quality control samples to the plate*

---

**Description**

A function to add QCs to plate. This function assumes adherence to ICH guideline M10 on bioanalytical method validation and study sample analysis Geneva, Switzerland (2022). If you are not following this guideline, you can set 'reg = TRUE' to ignore the restrictions.

**Usage**

```
add_QC(  
  plate,  
  lqc_conc,  
  mqc_conc,  
  hqc_conc,  
  extra = NULL,  
  n_qc = 3,  
  qc_serial = TRUE,  
  reg = TRUE,  
  group = NA  
)
```

**Arguments**

<code>plate</code>	PlateObj object
<code>lqc_conc</code>	low quality control concentration
<code>mqc_conc</code>	medium quality control concentration
<code>hqc_conc</code>	high quality control concentration
<code>extra</code>	numeric vector of extra QC concentrations.
<code>n_qc</code>	number of QC sets. Default is 3
<code>qc_serial</code>	logical. If TRUE, QCs are placed serially
<code>reg</code>	logical. Indicates if restrictions should not be applied to the QC samples. Default is TRUE
<code>group</code>	A string for bioanalytical group.

**Value**

PlateObj

---

add_samples	<i>Add samples to plate with pharmacokinetic attributes</i>
-------------	---

---

### Description

Add samples to plate with pharmacokinetic attributes

### Usage

```
add_samples(plate, samples, prefix = NA, dil = NA, group = NA, rep = 1)
```

### Arguments

plate	PlateObj
samples	A vector representing samples names. Must be unique.
prefix	A prefix to be added before samples names. Default is "Sub".
dil	A vector representing samples' dilution factor. Must be same length as samples.
group	A vector representing samples' bioanalytical group. Must be same length as samples.
rep	Number of technical replicates for each combination. Default is 1.

### Details

final name will be of form. Prefix-SampleName-Time-Concentration-Factor samples must be a unique vector and did not exist in the plate before. Time is either a vector or a single value. If it is a vector, it will be repeated for each sample. Conc, dil, factor and dose are either a vector or a single value. If it is a vector, it must be the corresponding length of samples.

Allowed routes are "IV", "IM", "IP", "SC", "PO", "INH" which are short for Intravenous, Intramuscular, Intraperitoneal, Subcutaneous, Per Os (oral), Inhalation.

Factor is an arbitrary factor used in the design like food vs fasted, healthy vs diseased, positive genotype ... etc.

### Value

PlateObj

### Examples

```
plate <- generate_96() |>  
add_samples(paste0("T", 1:12))
```

---

add_samples_db	<i>Add samples from the sample log to the plate</i>
----------------	---

---

**Description**

Add samples from the sample log to the plate

**Usage**

```
add_samples_db(plate, logIds, dil = 1, namestyle = 1, group = NA)
```

**Arguments**

plate	PlateObj
logIds	A vector of log IDs from the sample log.
dil	A vector with length corresponding number of logIds. See details.
namestyle	A numeric value indicating the naming style. 1 for long names, 2 for short names.
group	A string for bioanalytical group.

**Details**

This function will retrieve sample information from the sample log database using the provided log IDs. It constructs sample names based on the specified naming style and adds them to the plate. The 'dil' parameter allows specifying dilution factors for each sample, which will be appended to the sample names. If a single dilution factor is provided, it will be applied to all samples.

**Value**

PlateObj

---

add_samples_db2	<i>Add samples from the sample log to the plate with multiplication</i>
-----------------	---

---

**Description**

Add samples from the sample log to the plate with multiplication

**Usage**

```
add_samples_db2(plate, logIds, dil = c(1, 1), namestyle = 1, group = NA)
```

**Arguments**

plate	PlateObj
logIds	A vector of log IDs from the sample log.
dil	A vector with length corresponding number of repeats. See details.
namestyle	A numeric value indicating the naming style. 1 for long names, 2 for short names.
group	A string for bioanalytical group.

**Details**

This function is wrapper around 'add\_samples\_db()' that allows for quick replication of samples by dilution factor vector. For instance, if `dil = c(1,10)`, the samples will be repeated twice with one fold and 10 fold dilution factor each time.

---

add_suitability	<i>Add suitability sample to the plate</i>
-----------------	--

---

**Description**

Add suitability sample to the plate

**Usage**

```
add_suitability(plate, conc, label = "suitability", group = NA)
```

**Arguments**

plate	PlateObj object.
conc	numeric. Concentration of the suitability well.
label	character. Label for the suitability well. Default is "suitability".
group	A string for bioanalytical group.

**Value**

PlateObj

---

area\_report.PeakRes     *gt table of areas*

---

### Description

gt table of areas

### Usage

```
area_report.PeakRes(
  peaks_res,
  normalize = TRUE,
  blanks = TRUE,
  analytes = TRUE,
  standards = TRUE,
  QCs = TRUE,
  compounds = NULL
)
```

### Arguments

peaks_res	PeakRes object
normalize	logical. If TRUE, normalize the peak area by the IS area.
blanks	logical. If TRUE, include blanks
analytes	logical. If TRUE, include analytes
standards	logical. If TRUE, include standards
QCs	logical. If TRUE, include QCs
compounds	numeric vector of compound numbers to include. If NULL, include all compounds

---

calc\_var\_summary     *Calculate Summary Statistics for Each Concentration Level For Either Concentration, Area, or Area Ratio*

---

### Description

Calculate Summary Statistics for Each Concentration Level For Either Concentration, Area, or Area Ratio

**Usage**

```
calc_var_summary(
  df,
  col = "conc",
  acc_cutoff = 0.2,
  dev_cutoff = 0.2,
  type = "QC"
)
```

**Arguments**

df	Data frame with columns: stdconc (standardized concentration), conc (concentration), area (peak area), area_ratio (area ratio)
col	Column to calculate summary for ("conc", "area", or "area_ratio")
acc_cutoff	Accuracy threshold (default is 20%) for concentration vs standard concentration
dev_cutoff	Deviation threshold (default is 20%) for concentration vs standard concentration
type	Type of samples to include ("Standard", "QC", "DQC")

**Author(s)**

Omar I. Elashkar

---

check_chrom_cmpds	<i>Check Matching of Compound and Transitions in chrom_res and method database</i>
-------------------	--

---

**Description**

Check Matching of Compound and Transitions in chrom\_res and method database

**Usage**

```
check_chrom_cmpds(chrom_res, method_id)
```

**Arguments**

chrom_res	ChromRes object
method_id	Method ID in the method database This is important to give no error before merging quantification results to ensure consistency.

**Value**

TRUE if all compounds and transitions match, otherwise FALSE

---

chrom_app	<i>chrom_apps</i>
-----------	-------------------

---

**Description**

This function creates a shiny app for peak integration.

**Usage**

```
chrom_app()
```

---

combine_injec_lists	<i>Create Sample List with rigorous design</i>
---------------------	--

---

**Description**

Create Sample List with rigorous design

**Usage**

```
combine_injec_lists(  
  sample_lists,  
  n_equi = 10,  
  equi_pos,  
  equi_prefix = Sys.Date(),  
  equi_suffix = "equi",  
  equi_injec_vol = 0.5  
)
```

**Arguments**

sample_lists	a list of sample lists
n_equi	number of equilibration injections
equi_pos	position of equilibration injections. For format check details
equi_prefix	prefix for equilibration injections
equi_suffix	suffix for equilibration injections
equi_injec_vol	volume of equilibration injection

**Details**

The equi\_pos format will be Row:Column format. E.g: "A,1"

**Value**

InjecListObj object

---

combine_plates	<i>Combine plates in MultiPlate object</i>
----------------	--

---

**Description**

Combine plates in MultiPlate object

**Usage**

```
combine_plates(plates)
```

**Arguments**

plates	list of PlateObj objects
--------	--------------------------

**Value**

MultiPlate object

---

config_suitability	<i>Configure suitability runs</i>
--------------------	-----------------------------------

---

**Description**

Configure suitability runs by specifying vial position and range of runs to include.

**Usage**

```
config_suitability(quantres, vial_pos, start = NULL, end = NULL)
```

**Arguments**

quantres	QuantRes object
vial_pos	Vial position to use for suitability (e.g., "2:H,9")
start	Start position (1-based index) of runs to include. If NULL, starts from the first run.
end	End position (1-based index) of runs to include. If NULL, ends at the last run.

**Value**

Updated QuantRes object with suitability configuration.

---

<code>create_new_study</code>	<i>Create a new study in the database</i>
-------------------------------	---

---

**Description**

Create a new study in the database

**Usage**

```
create_new_study(df)
```

**Arguments**

<code>df</code>	A data frame with one row containing study details: <code>type</code> , <code>title</code> , <code>description</code> , <code>pkstudy</code> (logical), <code>subject_type</code>
-----------------	---

**Value**

A data frame with the created study details including generated id, `start_date`, and `status`

---

<code>cv</code>	<i>Calculate Coefficient of variation</i>
-----------------	---

---

**Description**

Calculate Coefficient of variation

**Usage**

```
cv(x, percent = TRUE)
```

**Arguments**

<code>x</code>	vector
<code>percent</code>	To return the value as percentage

**Details**

A simple calculation of the coefficient of variation (CV) is done as the standard deviation divided by the mean. By default, the result is in percentage.

**Value**

numeric

---

download_sample_list	<i>Download sample list from database to local spreadsheet with vendor specific format</i>
----------------------	--

---

**Description**

Download sample list from database to local spreadsheet with vendor specific format

**Usage**

```
download_sample_list(sample_list, vendor)
```

**Arguments**

sample_list	dataframe of sample list either from db or from write_injec_seq
vendor	currently only 'masslynx', 'masshunter' and 'analyst' are supported

**Details**

For all current vendors, the exported format will be in csv format, compatible with the respective software.

**Value**

dataframe

---

estim_dil_limit	<i>Estimate Dilution Limit Based on Additive and Proportional Errors and LLOQ</i>
-----------------	---

---

**Description**

Estimate Dilution Limit Based on Additive and Proportional Errors and LLOQ

**Usage**

```
estim_dil_limit(add_err, prop_err, lloq)
```

**Arguments**

add_err	Additive error (constant)
prop_err	Proportional error (CV)
lloq	Lower limit of quantification

**Author(s)**

Omar I. Elashkar

**Examples**

```
estim_dil_limit(add_err=0.1, prop_err=0.1, lloq=1)
estim_dil_limit(add_err=1, prop_err=0.1, lloq=55)
```

---

 estim\_lloq

*Estimate LLOQ From Existing Additive and Proportional errors*


---

**Description**

Estimate LLOQ From Existing Additive and Proportional errors

**Usage**

```
estim_lloq(add_err = 0.04, prop_err = 0.05, cv_lloq = 0.2, cv_lqc = 0.15)
```

**Arguments**

add_err	Additive error (constant)
prop_err	Proportional error (CV)
cv_lloq	Maximum coefficient of variation at LLOQ
cv_lqc	Maximum coefficient of variation at LQC

A method to estimate LLOQ from existing additive and proportional errors. The function does inequality constrained optimization to find the LLOQ.

**Author(s)**

Omar I. Elashkar

---

 export\_integration

*Export Expected RT*


---

**Description**

Export expected RT values for each peak in the chromatogram.

**Usage**

```
export_integration(chrom_res, path)
```

**Arguments**

chrom_res	ChromRes object
path	Path to save the file

---

export_pk_profiles	<i>Export PK profiles for a given compound in a specified format Currently supports "nonmem" format. The exported file will include a CSV with the PK data and an Excel file with the codebook.</i>
--------------------	---

---

**Description**

Export PK profiles for a given compound in a specified format Currently supports "nonmem" format. The exported file will include a CSV with the PK data and an Excel file with the codebook.

**Usage**

```
export_pk_profiles(x, compound_id, format = "NONMEM", filename = "data.zip")
```

**Arguments**

x	QuantRes object
compound_id	Compound ID for which to export PK profiles
format	Format to export (currently only "NONMEM" supported)
filename	Name of the output zip file (default: "data.zip")

**Author(s)**

Omar I. Elashkar

---

export_run	<i>Export run</i>
------------	-------------------

---

**Description**

Export run

**Usage**

```
export_run(peaks_res, path)
```

**Arguments**

peaks_res	PeakRes object
path	path to save csv

---

extract\_peak\_bounds     *Extract Peak Boundaries*

---

### Description

Extract peak boundaries for all samples for a given compound ID

### Usage

```
extract_peak_bounds(chrom_res, compound_id, samples_ids = NULL)
```

### Arguments

chrom_res	ChromRes object
compound_id	Compound ID
samples_ids	Sample IDs. If NULL, all samples will be used. The function automatically prioritizes observed peak boundaries (manual integration) over expected ones. If observed boundaries are not available, it falls back to expected boundaries.

### Value

Dataframe with compound\_id, min, max

---

fill\_scheme     *Filling orientation of the plate*

---

### Description

This function sets the filling scheme of the plate. The filling scheme is used to determine the order in which the samples are filled in the plate. The default filling scheme is horizontal, which means that the samples are filled from left to right and top to bottom. The vertical filling scheme means that the samples are filled from top to bottom and left to right.

### Usage

```
fill_scheme(  
  plate,  
  fill = "h",  
  tbound = "A",  
  bbound = "H",  
  lbound = 1,  
  rbound = 12  
)
```

**Arguments**

plate	PlateObj
fill	character. Filling scheme. Either "h" for horizontal, "v" for vertical.
tbound	character. Top bound of the filling scheme. Default is "A"
bbound	character. Bottom bound of the filling scheme. Default is "H"
lbound	numeric. Left bound of the filling scheme. Default is 1
rbound	numeric. Right bound of the filling scheme. Default is 12

**Value**

PlateObj

---

filter_chrom	<i>title Filter Chromatogram Peaks</i>
--------------	--

---

**Description**

This function filters chromatogram peaks based on transition ID and sample ID.

**Usage**

```
filter_chrom(
  chrom_res,
  transitions_ids = NULL,
  samples_id = NULL,
  compd_ids = NULL
)
```

**Arguments**

chrom_res	ChromRes object
transitions_ids	Vector of transition IDs to filter. If NULL, all transitions are returned.
samples_id	Sample ID to filter.
compd_ids	Compound ID to filter. It must be numeric. If NULL, all compounds are returned.

---

fit_var	<i>Estimate Additive and proportional errors from calibration data</i>
---------	--

---

**Description**

Estimate Additive and proportional errors from calibration data

**Usage**

```
fit_var(  
  data,  
  level = 0.95,  
  method = "nlminb",  
  bootstrap = FALSE,  
  n_boot = 1000  
)
```

**Arguments**

data	Data frame with columns: conc (concentration), stdconc (standardized concentration, e.g. conc/LLOQ)
level	Confidence level for the CI (default is 0.95)
method	Optimization method (default is "nlminb")
bootstrap	Logical indicating whether to perform bootstrap (default is TRUE)
n_boot	Number of bootstrap samples (default is 1000)

**Author(s)**

Omar I. Elashkar

---

formatted_print	<i>Format and print the results of fit_var</i>
-----------------	--

---

**Description**

Format and print the results of fit\_var

**Usage**

```
formatted_print(x, digits = 3)
```

**Arguments**

x	Data frame with results
digits	Number of digits to display

**Author(s)**

Omar I. Elashkar

---

generate\_96

*Generate 96 well plate*

---

**Description**

Generate 96 well plate

**Usage**

```
generate_96(descr = "", start_row = "A", start_col = 1)
```

**Arguments**

descr	plate description.
start_row	A letter corresponding to empty rows in a 96 well plate. Default is A.
start_col	A number indicating a column number to start with, given the start row. Default is 1.

Generate a typical 96 well plate. User need to specify the empty rows which a going to be used across the experiment.

**Value**

PlateObj

**Examples**

```
plate <- generate_96()
plot(plate)

plate <- generate_96("calibration", start_row = "C", start_col = 11)
plot(plate)
```

---

get_compound_ID	<i>Find Compound ID from compound Name</i>
-----------------	--

---

**Description**

This function returns the compound ID

**Usage**

```
get_compound_ID(chrom_res, compound_name)
```

**Arguments**

chrom_res	ChromRes object
compound_name	Compound Name

---

get_sample_ID	<i>Find Sample ID from sample Name</i>
---------------	--

---

**Description**

This function returns the sample ID

**Usage**

```
get_sample_ID(chrom_res, sample_name)
```

**Arguments**

chrom_res	ChromRes object
sample_name	Sample Name

---

get_sample_names	<i>Find sample names for all samples</i>
------------------	--

---

**Description**

Find sample names for all samples

**Usage**

```
get_sample_names(chrom_res)
```

**Arguments**

chrom_res	ChromRes object
-----------	-----------------

**Value**

data.frame with sample and sample\_id

---

has_default_bounds	<i>check if default expected RT is set for a compound</i>
--------------------	---

---

**Description**

check if default expected RT is set for a compound

**Usage**

```
has_default_bounds(chrom_res, compound_id)
```

**Arguments**

chrom_res	ChromRes object
compound_id	Compound ID

---

install_py_dep	<i>Install Python dependencies for PKbioanalysis</i>
----------------	--

---

**Description**

Install Python dependencies for PKbioanalysis

**Usage**

```
install_py_dep(..., envname = "PKbioanalysis")
```

**Arguments**

...	Additional arguments passed to <code>reticulate::py_install</code>
envname	Name of the virtual environment to create/use. Default is "PKbioanalysis"

**Value**

None

---

integrate	<i>integrate Peak with trapezoid method given start and end</i>
-----------	---

---

**Description**

integrate Peak with trapezoid method given start and end

**Usage**

```
integrate(chrom_res, compound_id, samples_ids, smoothed = TRUE)
```

**Arguments**

chrom_res	ChromRes object. Must have observed RT values
compound_id	Compound ID
samples_ids	Sample ID. If NULL, all samples will be used
smoothed	Logical. If TRUE, use smoothed chromatogram. Default is TRUE

---

is_integrated	<i>Check if peak was integrated for a specific compound</i>
---------------	---

---

**Description**

Check if peak was integrated for a specific compound

**Usage**

```
is_integrated(chrom_res, compound_id, sample_id = NULL)
```

```
## S4 method for signature 'ChromRes'
```

```
is_integrated(chrom_res, compound_id, sample_id = NULL)
```

```
## S4 method for signature 'ChromResBase'
```

```
is_integrated(chrom_res, compound_id, sample_id = NULL)
```

**Arguments**

chrom\_res      ChromRes or ChromResBase object

compound\_id    Compound ID

sample\_id      Sample ID. If NULL, all samples are checked

**Value**

logical

**Examples**

```
## Not run:
```

```
lapply(1:10, \(x) is_integrated(chrom_res, sample_id = 1, compound_id = 1))
```

```
## End(Not run)
```

---

is_smoothed	<i>Return an indicator if the chromatogram is smoothed</i>
-------------	--

---

**Description**

Return an indicator if the chromatogram is smoothed

**Usage**

```
is_smoothed(chrom_res)
```

**Arguments**

chrom\_res      ChromRes object

---

length, MultiPlate-method

*Length method for MultiPlate*

---

### Description

Length method for MultiPlate

### Usage

```
## S4 method for signature 'MultiPlate'  
length(x)
```

### Arguments

x                    MultiPlate object

### Value

number of plates

---

make\_calibration\_study

*Create a calibration study with calibration standards and QCs*

---

### Description

Create a calibration study with calibration standards and QCs

### Usage

```
make_calibration_study(  
  plate,  
  plate_std,  
  lqc_conc = NULL,  
  mqc_conc = NULL,  
  hqc_conc = NULL,  
  n_qc = NULL,  
  qc_serial = FALSE,  
  n_CS0IS0 = 1,  
  n_CS0IS1 = 2,  
  n_CS1IS0 = 1,  
  group = NA  
)
```

**Arguments**

plate	PlateObj object
plate_std	vector of calibration standards
lqc_conc	LQC concentration
mqc_conc	MQC concentration
hqc_conc	HQC concentration
n_qc	number of QC sets
qc_serial	logical. If TRUE, QCs are placed serially
n_CS0IS0	number of CS0IS0 (double) blanks
n_CS0IS1	number of CS0IS1 blanks
n_CS1IS0	number of CS1IS0 blanks
group	A string for bioanalytical group.

**Value**

PlateObj

---

make\_metabolic\_study *Create a metabolic study layout*

---

**Description**

Create a metabolic study layout

**Usage**

```
make_metabolic_study(
  study = "Metabolic Study",
  cmpds,
  time_points = c(0, 5, 10, 15, 30, 45, 60, 75, 90, 120),
  dose = NA,
  n_NAD = 3,
  n_noNAD = 2
)
```

**Arguments**

study	study name
cmpds	vector of compounds, including any standards
time_points	vector of time points
dose	dose amount. Default is NA
n_NAD	number of NAD positive samples. Default is 3
n_noNAD	number of NAD negative samples. Default is 2

**Details**

Note that this function does not require plate object. It will create a plate object automatically and return MultiPlate object

**Value**

MultiPlate object

---

nca_table	<i>Calculate Cmax, Tmax and AUC for each subject given a compound's PK profiles</i>
-----------	---

---

**Description**

Calculate Cmax, Tmax and AUC for each subject given a compound's PK profiles

**Usage**

```
nca_table(x, compound_id)
```

**Arguments**

x	QuantRes object with PK profiles extracted
compound_id	Compound ID for which to calculate NCA parameters

**Details**

This function calculates Cmax, Tmax and AUC for each subject given a compound's PK profiles.

**Value**

data frame with columns: subject\_id, cmax, tmax, auc\_last, compound\_id

---

pkmerge	<i>Merge PK profiles into QuantRes object</i>
---------	---

---

**Description**

Merge PK profiles into QuantRes object

**Usage**

```
pkmerge(x)
```

**Arguments**

x	QuantRes object
---	-----------------

---

plate_metadata	<i>Set plate description</i>
----------------	------------------------------

---

**Description**

Set plate description

**Usage**

```
plate_metadata(plate, descr)
```

**Arguments**

plate	PlateObj
descr	character. Description of the plate

**Value**

PlateObj

---

plate_tree	<i>Plot the design of the plate</i>
------------	-------------------------------------

---

**Description**

Plot the design of the plate

**Usage**

```
plate_tree(plate, plot = TRUE)
```

**Arguments**

plate	PlateObj object
plot	logical. If TRUE, plot the tree

**Value**

data.tree Node object or DiagrammeR object plot\_tree will focus only on bioanalytical vial types, namely blanks, analytes, standards, QCs. The tree order will be plate\_id, then group, then vial type, then entity, then number of technical replicates.

---

plot.PlateObj                      *Plotting 96 well plate*

---

## Description

Plotting 96 well plate

## Usage

```
## S3 method for class 'PlateObj'
plot(
  x,
  color = "conc",
  Instrument = "",
  caption = "",
  label_size = 1,
  transform_dil = FALSE,
  watermark = "auto",
  layoutOverlay = FALSE,
  path = NULL,
  ...
)
```

## Arguments

x	PlateObj
color	character. Coloring variable. Choices: "conc", "group", "dil", "study", "time", "factor", "samples", "arm", "sex", "dose", "route", "matrix". Default is "conc"
Instrument	A string placed at subtitle
caption	A string place at plate caption
label_size	numeric. Size of the label. Default is 15
transform_dil	logical. If TRUE, transform the dilution factor to the label
watermark	character. If "auto", a watermark is added to the plot. If "none", no watermark is added. Default is "auto"
layoutOverlay	logical. If TRUE, overlay the plot layout. Default is FALSE
path	If not null, must be a path to save plate image
...	additional arguments passed to ggplot2::ggsave

## Value

ggplot object

**Examples**

```

plate <- generate_96("new_plate", "C", 11) |>
  add_blank(IS = FALSE, analyte = FALSE) |>
  add_blank(IS = TRUE, analyte = FALSE) |>
  add_samples(c(
    "RD_per1", "RD_in1", "RD_T30", "RD_T60", "RD_T90", "RD_per2", "RD_in2",
    "EE_in0", "EE_T30", "EE_in30", "EE_T60", "EE_in60", "EE_T90", "EE_in90"
  ))
plot(plate)

```

---

plot\_chrom

*Plot Chromatogram per Sample for Selected transitions*


---

**Description**

This function plots chromatograms for selected transitions per sample.

**Usage**

```

plot_chrom(
  chrom_res,
  ncol = 2,
  transitions_ids = NULL,
  sample_id,
  integrated = FALSE,
  show_RT = FALSE,
  smoothed = FALSE
)

```

**Arguments**

chrom_res	ChromRes object
ncol	Number of columns for facet_wrap. If 0, the chromatograms are overlaid in a single plot.
transitions_ids	Vector of transition IDs to plot. If NULL, all transitions are plotted.
sample_id	Sample ID to plot.
integrated	Boolean to show integrated area overlaid
show_RT	Boolean to show RT values
smoothed	Boolean to show smoothed chromatogram

**Examples**

```
## Not run:
path <- system.file("extdata", "waters_raw_ex", package="PKbioanalysis")
main <- read_chrom(path, method = 1)
plot_chrom(main, ncol = 2, transitions_ids = c(18,19,20), sample_id = 3)
plot_chrom(main, ncol = 3, transitions_ids = c(18,19,20), sample_id = 3)
plot_chrom(main, ncol = 1, transitions_ids = c(18,19,20), sample_id = 3)
plot_chrom(main, ncol = NULL, transitions_ids = c(18,19,20), sample_id = 3)

## End(Not run)
```

---

plot\_peak\_areas.PeakRes

*Plot peak areas*

---

**Description**

Plot peak areas

**Usage**

```
plot_peak_areas.PeakRes(
  peaks_res,
  normalize = TRUE,
  blanks = TRUE,
  compounds = NULL,
  analytes = TRUE,
  standards = TRUE,
  QCs = TRUE,
  type = "bar"
)
```

**Arguments**

peaks_res	PeakRes object
normalize	logical. If TRUE, normalize the peak area by the IS area.
blanks	logical. If TRUE, plot blanks
compounds	numeric vector of compound numbers to include. If NULL, include all compounds
analytes	logical. If TRUE, plot analytes
standards	logical. If TRUE, plot standards
QCs	logical. If TRUE, plot QCs
type	character. Either "bar" or "line"

**Value**

ggplot2 object

---

plot_RT.ChromRes	<i>Plotting RT intervals of chromatogram</i>
------------------	--

---

**Description**

Plotting RT intervals of chromatogram

**Usage**

```
plot_RT.ChromRes(chrom_res)
```

**Arguments**

chrom_res	ChromRes object
-----------	-----------------

---

plot_RT.PeakRes	<i>Plot RT</i>
-----------------	----------------

---

**Description**

Plot RT

**Usage**

```
plot_RT.PeakRes(
  peaks_res,
  normalize = TRUE,
  blanks = TRUE,
  analytes = TRUE,
  standards = TRUE,
  QCs = TRUE,
  facet = FALSE,
  compounds = NULL
)
```

**Arguments**

peaks_res	PeakRes object
normalize	logical. If TRUE, normalize the peak area by the IS area.
blanks	logical. If TRUE, plot blanks
analytes	logical. If TRUE, plot analytes
standards	logical. If TRUE, plot standards
QCs	logical. If TRUE, plot QCs
facet	logical. If TRUE, facet by compound name
compounds	numeric vector of compound numbers to include. If NULL, include all compounds

**Value**

ggplot2 object

---

plot_var_pattern	<i>Plot Relationship Between Concentration and CV/SD</i>
------------------	--

---

**Description**

Plot Relationship Between Concentration and CV/SD

**Usage**

```
plot_var_pattern(df, title = "")
```

**Arguments**

df	Data frame with columns: stdconc (standardized concentration), cv (coefficient of variation), sdev (standard deviation), Type (e.g., "Estimated", "Observed")
title	Plot title

**Author(s)**

Omar I. Elashkar

---

precision_per_vial	<i>Precision per vial</i>
--------------------	---------------------------

---

**Description**

Precision per vial

**Usage**

```
precision_per_vial(peaks_res, suitability = FALSE)
```

**Arguments**

peaks_res	PeakRes object
suitability	logical. If TRUE, suitability samples are ignored

**Value**

ggplot2 object

---

prefilter\_precision\_data  
*Filter data*

---

**Description**

Filter data

**Usage**

```
prefilter_precision_data(  
  x,  
  type,  
  acc_cutoff = 0.2,  
  dev_cutoff = 0.2,  
  compound_id = NULL  
)  
  
## S4 method for signature 'QuantRes'  
prefilter_precision_data(  
  x,  
  type,  
  acc_cutoff = 0.2,  
  dev_cutoff = 0.2,  
  compound_id = NULL  
)  
  
## S4 method for signature 'data.frame'  
prefilter_precision_data(x, type, acc_cutoff = 0.2, dev_cutoff = 0.2)
```

**Arguments**

x	Dataframe or QuantRes Object
type	QC, DQC, or Standard
acc_cutoff	Accuracy cutoff. 20% by default
dev_cutoff	Deviation cutoff. 20% by default
compound_id	Compound ID to filter. If NULL, all compounds are considered

**Value**

Filtered data

**Author(s)**

Omar I. Elashkar

---

quant_app	<i>Quantification App</i>
-----------	---------------------------

---

### Description

This function creates a shiny app for quantification after peak integration

### Usage

```
quant_app()
```

---

read_chrom	<i>Read Chromatogram Files</i>
------------	--------------------------------

---

### Description

This function reads chromatogram files from a directory and returns a data frame with the chromatogram data.

### Usage

```
read_chrom(dir, format = "waters_raw", method)
```

### Arguments

dir	directory for chromatograms
format	format of the chromatogram files. Options are "waters_raw" and "mzML".
method	LC-MS/MS method ID saved available in the database.

### Examples

```
## Not run:  
path <- system.file("extdata", "waters_raw_ex", package="PKbioanalysis")  
main <- read_chrom(path, method = 1)  
  
## End(Not run)
```

---

read\_experiment\_results  
*Read experiment results*

---

**Description**

Read experiment results

**Usage**

```
read_experiment_results(  
  x,  
  drop_prefix = FALSE,  
  vendor = "targetlynx_xml",  
  logkey = "Index"  
)
```

**Arguments**

x	path to experiment results. See details
drop_prefix	logical. If TRUE, drop the prefix from the sample name
vendor	vendor name. Currently only "targetlynx_xml" or "targetlynx_csv" are supported.
logkey	character. The column name in the targetlynx CSV file that contains the injection sequence or log key. Default is "Index" which is the default column in targetlynx CSV exports, but it can be customized if the user has a different column name for the injection sequence.

**Details**

Currently only targetlynx XML or CSV exported files are supported.

**Value**

QuantRes object with the results of the experiment.

---

register\_plate      *This will save the plate to the database*

---

**Description**

This will save the plate to the database

**Usage**

```
register_plate(plate)
```

**Arguments**

plate            PlateObj object or MultiPlate object

**Value**

PlateObj object or list of PlateObj objects

---

response_to_conc	<i>Convert response to concentration</i>
------------------	--

---

**Description**

Convert response to concentration

**Usage**

```
response_to_conc(quantres, compound_id, response)
```

**Arguments**

quantres        QuantRes object  
compound\_id    character  
response        numeric. Must match the response type used in linearity. Either abs\_response or rel\_response

**Value**

numeric

---

reverse_predict	<i>Reverse predict concentration from response</i>
-----------------	--

---

**Description**

Reverse predict concentration from response

**Usage**

```
reverse_predict(fit, newdata, intercept)
```

**Arguments**

fit	lm object
newdata	vector or data frame with response values
intercept	logical. Whether the model has intercept or not

**Value**

numeric. Estimated concentration

**Author(s)**

Omar I. Elashkar

---

run_summary	<i>Get Summary of an object</i>
-------------	---------------------------------

---

**Description**

Get Summary of an object

**Usage**

```
run_summary(object)

## S3 method for class 'PeakRes'
run_summary(object)
```

**Arguments**

object	A PeakRes object
--------	------------------

---

smooth_chrom	<i>Smooth Chromatogram Peaks</i>
--------------	----------------------------------

---

**Description**

This function smooths chromatogram peaks using different algorithms.

**Usage**

```
smooth_chrom(chrom_res, filter = "mean", window = 2, iter = 2)
```

**Arguments**

chrom_res	ChromRes object
filter	Filter to use. Options are "mean", "median", "savgol", "gaussian"
window	Window size for the filter
iter	Number of iterations. If 0, no smoothing is applied.

---

study_app	<i>bioanalytic_app</i>
-----------	------------------------

---

**Description**

This function creates a shiny app for plate management

**Usage**

```
study_app()
```

**Value**

A shiny app. No default return value. Can return a PlateObj if reuse\_plate\_button is clicked

---

update_RT	<i>Manually Update Observed RT for either all compounds, all next samples, or single compound and sample</i>
-----------	--

---

**Description**

Update RT for either all compounds, all next samples, or single compound and sample

**Usage**

```
update_RT(
  chrom_res,
  compound_id,
  sample_id = NULL,
  peak_start,
  peak_end,
  mode = "auto",
  target = "single",
  force = FALSE,
  comment = "",
  flag = FALSE
)
```

**Arguments**

chrom_res	ChromRes object
compound_id	Compound ID
sample_id	Sample ID (required for "single" and "all_next", must be NULL for "all")
peak_start	Minimum RT value
peak_end	Maximum RT value
mode	Mode of update. Options are "auto", "manual", "ai". Default is "auto"
target	Target of update. Options are "single", "all", "all_next". Default is "single"
force	Force update if previous peak exists. Default is FALSE
comment	Comment for the update. Default is an empty string
flag	Flag the peak after update. Default is FALSE

**Details**

- target = "single": Updates RT for one compound and sample - target = "all": Updates expected RT for all samples (sets expected bounds) - target = "all\_next": Updates RT for specified sample and subsequent samples

All modes affect both observed and expected RT values: - "manual": Sets exact peak bounds, marks as manual - "auto": Auto-detects peaks within bounds - "ai": AI-based peak detection

**Value**

Updated ChromRes object

**Examples**

```
## Not run:
update_RT(chrom_res, compound_id = 1, sample_id = 1,
          peak_start = 0.1, peak_end = 1, target = "single")

## End(Not run)
```

---

write_injec_seq	<i>Write injection sequence to database</i>
-----------------	---

---

**Description**

Write injection sequence to database

**Usage**

```
write_injec_seq(injec_seq)
```

**Arguments**

injec\_seq      InjecListObj object

**Value**

dataframe

---

[[,MultiPlate-method    *Subsetting method for MultiPlate*

---

**Description**

Subsetting method for MultiPlate

**Usage**

```
## S4 method for signature 'MultiPlate'  
x[[i, j, ...]]
```

**Arguments**

x                  MultiPlate object  
i                  index  
j                  index  
...                additional arguments

**Value**

PlateObj object

# Index

`[[`,MultiPlate-method, 42

`add_blank`, 3  
`add_cs_curve`, 4  
`add_DB`, 5  
`add_DQC`, 5  
`add_QC`, 6  
`add_samples`, 7  
`add_samples_db`, 8  
`add_samples_db2`, 8  
`add_suitability`, 9  
`area_report.PeakRes`, 10

`calc_var_summary`, 10  
`check_chrom_cmpds`, 11  
`chrom_app`, 12  
`combine_injec_lists`, 12  
`combine_plates`, 13  
`config_suitability`, 13  
`create_new_study`, 14  
`cv`, 14

`download_sample_list`, 15

`estim_dil_limit`, 15  
`estim_lloq`, 16  
`export_integration`, 16  
`export_pk_profiles`, 17  
`export_run`, 17  
`extract_peak_bounds`, 18

`fill_scheme`, 18  
`filter_chrom`, 19  
`fit_var`, 20  
`formatted_print`, 20

`generate_96`, 21  
`get_compound_ID`, 22  
`get_sample_ID`, 22  
`get_sample_names`, 23

`has_default_bounds`, 23

`install_py_dep`, 24  
`integrate`, 24  
`is_integrated`, 25  
`is_integrated`,ChromRes-method  
    (`is_integrated`), 25  
`is_integrated`,ChromResBase-method  
    (`is_integrated`), 25  
`is_smoothed`, 25

`length`,MultiPlate-method, 26

`make_calibration_study`, 26  
`make_metabolic_study`, 27

`nca_table`, 28

`pkmerge`, 28  
`plate_metadata`, 29  
`plate_tree`, 29  
`plot.PlateObj`, 30  
`plot_chrom`, 31  
`plot_peak_areas.PeakRes`, 32  
`plot_RT.ChromRes`, 33  
`plot_RT.PeakRes`, 33  
`plot_var_pattern`, 34  
`precision_per_vial`, 34  
`prefilter_precision_data`, 35  
`prefilter_precision_data`,data.frame-method  
    (`prefilter_precision_data`), 35  
`prefilter_precision_data`,QuantRes-method  
    (`prefilter_precision_data`), 35

`quant_app`, 36

`read_chrom`, 36  
`read_experiment_results`, 37  
`register_plate`, 37  
`response_to_conc`, 38  
`reverse_predict`, 38

run\_summary, [39](#)

smooth\_chrom, [39](#)

study\_app, [40](#)

update\_RT, [40](#)

write\_injec\_seq, [41](#)