Package: eCerto (via r-universe)

November 20, 2026

Type Package

Title Statistical Tests for the Production of Reference Materials

Version 0.8.30

Date 2026-11-20

Maintainer Jan Lisec <jan.lisec@bam.de>

Description The production of certified reference materials (CRMs) requires various statistical tests depending on the task and recorded data to ensure that reported values of CRMs are appropriate. Often these tests are performed according to the procedures described in 'ISO GUIDE 35:2017'. The 'eCerto' package contains a 'Shiny' app which provides functionality to load, process, report and backup data recorded during CRM production and facilitates following the recommended procedures. It is described in Lisec et al (2023) <doi:10.1007/s00216-023-05099-3> and can also be accessed online <https://apps.bam.de/shn00/eCerto/> without package installation.

URL https://github.com/janlisec/eCerto

BugReports https://github.com/janlisec/eCerto/issues

License MIT + file LICENSE

Encoding UTF-8

Language en-US

LazyData true

LazyDataCompression bzip2

Depends R (>= 4.1.0)

Imports bslib, config, DT, flextable, golem, knitr, moments, openxlsx, purrr, R6, rmarkdown (>= 1.5), shiny, shinyjs, shinyWidgets, tidyxl, xml2

Suggests covr, curl, fs, htmltools, jsonlite, rlang, testthat (>= 3.0.0), vdiffr, webshot2

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Jan Lisec [cre, aut] (ORCID:
<https://orcid.org/0000-0003-1220-2286>), Frederik Kreß [ctb]

Repository https://cran.r-universe.dev

Date/Publication 2026-11-20 13:40:02 UTC

RemoteUrl https://github.com/cran/eCerto

RemoteRef HEAD

RemoteSha 5a58060a2f9ae178d454a8490bb9806de47db697

Contents

assert_col 2
BAMlogo_raster 4
calc_time_diff 4
CRM001 5
cvals_Dixon 6
cvals_Grubbs2 6
eCerto_R6Class 7
ldply_base 10
LTS001 11
markdown2expression 12
run_app 13
save_link 14
steyx 14

Index **15**

`assert_col` *Assert a specific column (type and position) in a data frame.*

Description

`assert_col` will check in a data.frame for name, position, type of a specific column and ensure that the return value (data frame) contains a respective column. If possible, the current values are converted into the specified type.

Usage

```
assert_col(
  df,
  name,
  pos = NULL,
  type = c("character", "integer", "numeric", "factor", "logical", "Date"),
  fuzzy_name = TRUE,
  default_value = NULL
)
```

Arguments

df	Input data frame.
name	Name of the column to ensure (and to search for).
pos	Position of this column. NULL to keep position where found in df.
type	Desired data type of this column.
fuzzy_name	Allow fuzzy matching (additional blanks and case insensitive search allowed).
default_value	Default value if column needs to be created or can not be converted to specified type. Keep NULL to use pre defined default values.

Details

tbd.

Value

A data frame with a column of the specified name and type at the specified position. An error message is attached to the result as an attribute in case of unexpected events.

Examples

```
x <- data.frame(
  "analyte" = c("A", "B"),
  "tmp" = rep(0L, 2),
  "unit" = c("x", "y")
)
str(x)
ac <- eCerto::assert_col
str(ac(df = x, name = "analyte", pos = 1, type = "factor"))
str(ac(df = x, name = "Analyte", pos = 3, type = "character"))
str(ac(df = x, name = " Analyte", pos = 2, type = "factor"))
str(ac(df = x, name = "Analyte", pos = 2, type = "factor", fuzzy_name = FALSE))
str(ac(df = x, name = "test", type = "factor", default_value = "test"))
# this will lead to NAs in column unit because the conversion does not lead to an error
# hence the default value is not used
str(ac(df = x, name = "unit", type = "numeric", default_value = 10))
# this will lead to the specified default data in column unit because the
# conversion attempt does lead to an error
str(ac(df = x, name = "unit", type = "Date"))
```

```
str(ac(df = data.frame("test" = "2022-03-31"), name = "test", type = "Date"))

# show type and class of internal default values
x <- data.frame(
  "character" = "", "integer" = 0L, "numeric" = 0, "factor" = factor(NA),
  "logical" = NA, "date" = Sys.Date(), NA
)
sapply(1:ncol(x), function(i) {
  typeof(x[, i])
})
sapply(1:ncol(x), function(i) {
  class(x[, i])
})
```

`BAMlogo_raster` *A raster representation of the BAM logo to be used in Reports.*

Description

A raster representation of the BAM logo to be used in Reports.

Usage

```
data(BAMlogo_raster)
```

Format

A raster representation of the BAM logo png file.

Source

jan.lisec@bam.de

`calc_time_diff` *Calculate time differences for suitable vectors.*

Description

Calculation of a time difference between time points in a vector `x` and a specific start date `d_start` in month (days or years).

Usage

```
calc_time_diff(
  x = NULL,
  d_start = NULL,
  type = c("mon", "day", "year"),
  origin = "1900-01-01",
  exact = FALSE
)
```

Arguments

x	A vector of dates or character in format 'yyyy-mm-dd'.
d_start	A specific start date (if unspecified the minimum of x will be used to ensure positive values).
type	You may specify 'year' or 'day' instead of month here.
origin	The origin used.
exact	Function will return exact values instead of full month and year if this is set to TRUE.

Value

A numeric vector of length x containing calculated time differences in the unit specified by type. Not a difftime object.

Examples

```
x <- c("2022-02-01", "2022-02-03", "2022-03-01", "2024-02-01")
calc_time_diff(x = x)
calc_time_diff(x = x, exact = TRUE)
calc_time_diff(x = x, type = "day")
calc_time_diff(x = x, type = "year")
calc_time_diff(x = x, type = "year", d_start = "2021-12-31")
calc_time_diff(x = 1:3, type = "day", origin = Sys.Date())
```

CRM001 *An example set of data collected for a CRM.*

Description

An example set of data collected for a CRM.

Usage

```
data(CRM001)
```

Format

A list of length = 6 containing CRM test data.

Source

jan.lisec@bam.de

cvals_Dixon *Dixon critical values table.*

Description

Dixon critical values table.

Usage

```
data(cvals_Dixon)
```

Format

A data frame containing Dixon critical values with n in rows and alpha in cols.

Source

http://www.statistics4u.com/fundstat_eng/cc_outlier_tests_dixon.html

cvals_Grubbs2 *Grubbs2 critical values table.*

Description

Grubbs2 critical values table.

Usage

```
data(cvals_Grubbs2)
```

Format

A data frame containing critical values for Double Grubbs test with n in rows and alpha in cols.

Source

`outliers` package and https://link.springer.com/article/10.1007/s10182-011-0185-y.

`eCerto_R6Class` *A reactive class based on an R6 object.*

Description

Builds a class, which allows only restricted access to the contained 'reactiveValues'. Elements should be accessed via `getValue()`. Possible advantages are that (1) structure of 'reactiveValues' is clear from the beginning (no function like "addVariable" should exist!) and that (2) functions to calculate the mean or plot current data can be implemented here directly.

General access to data object (so data object can maybe get changed without that much code edit)

Returns element. If 'key' is used, reactivity not working correctly. Preferable way for calling `getValue(df, key)`, see example

Usage

```
setValue(df, key, value)

getValue(df, key = NULL)
```

Arguments

`df`	An object of class R6.
`key`	Key value within R6 object 'df'.
`value`	Value to set.

Value

Nothing. The R6 object is updated automatically.

Value of 'key' from 'df'.

Active bindings

`cur_an` Set or return the current analyte (reactiveVal) via an active binding.

Methods

Public methods:

- `eCerto$new()`
- `eCerto$get()`
- `eCerto$set()`
- `eCerto$c_plot()`
- `eCerto$c_lab_means()`
- `eCerto$c_analytes()`
- `eCerto$c_lab_codes()`
- `eCerto$a_p()`

- eCerto$e_present()
- eCerto$c_fltData()
- eCerto$clone()

`eCerto$new()`: Write the (reactive) value of element 'keys' from list 'l'.

Usage:

```
eCerto$new(rv)
```

Arguments:

`rv` 'reactiveValues' object.

Returns: A new 'eCerto' object.

`eCerto$get()`: Read the value of field element of R6 object.

Usage:

```
eCerto$get(keys = NULL)
```

Arguments:

`keys` Name of list element.

Returns: Current value of field.

`eCerto$set()`: Set element of R6 object defined by 'keys' to new value.

Usage:

```
eCerto$set(keys = NULL, value)
```

Arguments:

`keys` Name(s) of list element.
`value` New value.

Returns: New value of element (invisible).

`eCerto$c_plot()`: Plot the certification data either provided by the user or from the private slot of self.

Usage:

```
eCerto$c_plot(data, annotate_id = FALSE, filename_labels = FALSE)
```

Arguments:

`data` data.frame containing columns 'value', 'Lab' and 'L_flt' for a specific analyte.
`annotate_id` T/F to overlay the plot with ID as text if column 'ID' is present.
`filename_labels` T/F to use imported file names as labels on x-axes.

Returns: A plot.

`eCerto$c_lab_means()`: Compute the analyte means for a data set filtered for a specific analyte.

Usage:

```
eCerto$c_lab_means(data)
```

Arguments:

`data` data.frame containing columns 'analyte', 'value', 'Lab', 'S_flt' and 'L_flt'.

Returns: A data.frame of lab means.

`eCerto$c_analytes()`: Return analyte names currently in apm.

Usage:

```
eCerto$c_analytes()
```

Returns: A named character vector.

`eCerto$c_lab_codes()`: Return lab codes currently in C data.

Usage:

```
eCerto$c_lab_codes()
```

Returns: A named character vector.

`eCerto$a_p()`: Return currently specified values of a type for all analytes.

Usage:

```
eCerto$a_p(
  val = c("precision", "precision_export", "pooling", "confirmed", "unit", "name")
)
```

Arguments:

`val` A character value indicating the item of the apm list to be extracted

Returns: A named vector.

`eCerto$e_present()`: Return modules with existing data.

Usage:

```
eCerto$e_present()
```

Returns: A named logical vector.

`eCerto$c_fltData()`: Filter the full data set for a specific analyte and remove all 'S_flt' but keep 'L_flt'.

Usage:

```
eCerto$c_fltData(recalc = FALSE)
```

Arguments:

`recalc` If TRUE triggers a recalculation and returns current object if FALSE..

Returns: A data.frame with filtered data of a single analyte.

`eCerto$clone()`: The objects of this class are cloneable with this method.

Usage:

```
eCerto$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if (interactive()) {
  # establish new Shiny session and new eCerto object
  ShinySession <- shiny::MockShinySession$new()
  test <- eCerto::eCerto$new()
  # view current value stored in specific eCerto slot and register observer
  shiny::isolate(eCerto::getValue(test, c("Certification", "data")))
  shiny::observeEvent(eCerto::getValue(test, c("Certification", "data")), {
    message("Certification$data changed:", eCerto::getValue(test, "Certification")$data)
  })
  # change value of specific eCerto slot and flush reactivity to trigger observer
  shiny::isolate(eCerto::setValue(test, c("Certification", "data"), 5))
  ShinySession$flushReact()
  shiny::isolate(eCerto::getValue(test, c("Certification", "data")))
}
tmp <- eCerto$new()
shiny::isolate(tmp$c_plot())
shiny::isolate(tmp$c_lab_means())
tmp$c_analytes()
tmp$c_lab_codes()
tmp$a_p()
tmp$a_p("pooling")
ca <- shiny::isolate(tmp$cur_an)
tmp$a_p("pooling")[ca]
shiny::isolate(tmp$e_present())
tmp$c_fltData()
shiny::isolate(tmp$cur_an <- "Fe")
shiny::isolate(tmp$cur_an)
tmp$c_fltData()
x <- shiny::isolate(eCerto::getValue(tmp, c("General", "apm")))
x[[shiny::isolate(tmp$cur_an)]][["lab_filter"]] <- "L2"
shiny::isolate(eCerto::setValue(tmp, c("General", "apm"), x))
tmp$c_fltData()
tmp$c_fltData(recalc = TRUE)

# Only run examples in interactive R sessions
if (interactive()) {
  rv <- eCerto$new(init_rv())
  setValue(rv, c("Certification", "data"), 5)
  getValue(rv, c("Certification", "data")) # is 5?
  setValue(rv, c("General", "user"), "Franz")
  getValue(rv, c("General", "user"))
}
```

`ldply_base` *ldply_base*

Description

A base R implementation of plyr::ldply

Usage

```
ldply_base(.data, .fun = identity, .progress = "none", .id = NA, ...)
```

Arguments

.data	A list or vector.
.fun	Function to apply to each item.
.progress	Show progress bar if 'text'.
.id	Name of the index column (used if .data is a named list). Pass NULL to avoid creation of the index column. For compatibility, omit this argument or pass NA to avoid converting the index column to a factor; in this case, ".id" is used as colum name.
...	Arguments to .fun.

Examples

```
x <- list(a = data.frame(x = 1:2, y = 5:6), b = data.frame(x = 3:4, y = 7:8))
ldply_base(x)
ldply_base(x, .id = NULL)
ldply_base(unname(x))
ldply_base(x, .id = "test")
# compare against standard plyr::ldply
#plyr::ldply(x, .id="test")
str(ldply_base(x))
#str(plyr::ldply(x))
x <- c("01.01.2025","02.01.2025")
ldply_base(x, as.Date.character, tryFormats = "%d.%m.%Y")
#plyr::ldply(x, as.Date.character, tryFormats = "%d.%m.%Y")
```

LTS001 *An example set of data collected for a LTS monitoring.*

Description

An example set of data collected for a LTS monitoring.

Usage

```
data(LTS001)
```

Format

A list of lists of length = 2 containing LTS test data.

Source

jan.lisec@bam.de

`markdown2expression` *Convert Markdown String to R Plotmath Expression*

Description

Parses a character string containing markdown-style formatting syntax and converts it into an R plotmath expression suitable for use in plot labels and annotations.

Usage

```
markdown2expression(x, vectorize = FALSE)
```

Arguments

`x`	A character vector. Only the first element is used.
`vectorize`	Set TRUE to process all elements of x returning a list of expressions.

Details

The following markdown syntax is supported:

- `**text**` — bold
- `*text*` — italic
- `base^sup^` — superscript
- `base~sub~` — subscript

Formatting can be nested (e.g., bold inside superscript). The returned expression can be passed directly to `plot()` arguments such as `main`, `xlab`, or `ylab`, or to `expression()`.

Value

A language object (R expression) suitable for use in plotting functions that support plotmath.

See Also

`plotmath`

Examples

```
markdown2expression("**bold text**")
markdown2expression("*italic text*")
markdown2expression("x^2^")
markdown2expression("H~2~O")
markdown2expression("**H~2~O**")
plot(1,1, main = markdown2expression("R^2^ = 0.99"), pch=".")
text(1,1, markdown2expression("Water = **H~2~O**"))
```

run_app *Run the Shiny Application*

Description

Run the Shiny Application

Usage

```
run_app(
  onStart = NULL,
  options = list(port = 3838),
  enableBookmarking = NULL,
  uiPattern = "/",
  ...
)
```

Arguments

onStart	A function that will be called before the app is actually run. This is only needed for shinyAppObj, since in the shinyAppDir case, a global.R file can be used for this purpose.
options	Named options that should be passed to the runApp call (these can be any of the following: "port", "launch.browser", "host", "quiet", "display.mode" and "test.mode"). You can also specify width and height parameters which provide a hint to the embedding environment about the ideal height/width for the app.
enableBookmarking	Can be one of "url", "server", or "disable". The default value, NULL, will respect the setting from any previous calls to enableBookmarking(). See enableBookmarking() for more information on bookmarking your app.
uiPattern	A regular expression that will be applied to each GET request to determine whether the ui should be used to handle the request. Note that the entire request path must match the regular expression in order for the match to be considered successful.
...	arguments to pass to golem_opts. See ?golem::get_golem_options for more details.

save_link *save_link.*

Description

Provide save link tag

Usage

```
save_link(text, link)
```

Arguments

text	text.
link	link.

Examples

```
save_link("KS", "https://rdrr.io/r/stats/ks.test.html")
```

steyx *Implementation of the STEYX function from Excel.*

Description

Translation of STEYX function from Excel to R. It is implemented according to the formula described in https://support.microsoft.com/en-us/office/steyx-function-6ce74b2c-449d-4a6e-b9ac-f9cef5ba48ab. At least 3 finite pairs of data points are required for the calculation.

Usage

```
steyx(x, y)
```

Arguments

x	x values as numeric vector.
y	y values as numeric vector of similar length as x.

Value

The standard error of the predicted y-value for each x in the regression.

Examples

```
steyx(x = 1:3, y = 2:4)
steyx(x = 1:3, y = c(2, 3.1, 3.9))
```

Index

∗ **datasets**
    BAMlogo_raster, 4
    CRM001, 5
    cvals_Dixon, 6
    cvals_Grubbs2, 6
    LTS001, 11

assert_col, 2

BAMlogo_raster, 4

calc_time_diff, 4
CRM001, 5
cvals_Dixon, 6
cvals_Grubbs2, 6

eCerto *(*eCerto_R6Class*)*, 7
eCerto_R6Class, 7
enableBookmarking(), *13*

getValue *(*eCerto_R6Class*)*, 7
getValue(), *7*

ldply_base, 10
LTS001, 11

markdown2expression, 12

plotmath, *12*

run_app, 13

save_link, 14
setValue *(*eCerto_R6Class*)*, 7
steyx, 14