Package 'eCerto'

March 28, 2025

Type Package

Title Statistical Tests for the Production of Reference Materials

Version 0.8.5

Date 2025-03-28

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, dplyr, DT, golem, knitr, magick, markdown (>= 2.0), moments, openxlsx, plyr, purrr, R6, rmarkdown (>= 1.5), shiny, shinyjs, shinyWidgets, tidyxl, xml2

Suggests covr, curl, fs, jsonlite, rlang, shinytest2, testthat (>= 3.0.0), vdiffr, webshot2

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation no

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

Repository CRAN

Date/Publication 2025-03-28 13:00:02 UTC

Contents

`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])
})
```

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()`

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

Usage:

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

Arguments:

`rv` 'reactiveValues' object.

Returns: A new 'eCerto' object.

Method `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.

Method `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).

Method `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.

Method `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.

Method `c_analytes()`**:** Return analyte names currently in apm.

Usage:

```
eCerto$c_analytes()
```

Returns: A named character vector.

Method `c_lab_codes()`**:** Return lab codes currently in C data.

Usage:

```
eCerto$c_lab_codes()
```

Returns: A named character vector.

Method `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.

Method `e_present()`**:** Return modules with existing data.

Usage:

```
eCerto$e_present()
```

Returns: A named logical vector.

Method `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.

Method `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"))
}
```

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

---

run_app *Run the Shiny Application*

Description

Run the Shiny Application

Usage

```
run_app(
  onStart = NULL,
  options = list(),
  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.

`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 `http://office.microsoft.com/en-au/excel-help/steyx-function-HP010062545.aspx`. 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