

# Package ‘kdry’

April 18, 2023

**Title** K's ``Don't Repeat Yourself'-Collection

**Version** 0.0.1

**Description** A personal collection of helper functions and code snippets to avoid redundancy in the spirit of the ``Don't repeat yourself" principle of software development  
([https://en.wikipedia.org/wiki/Don%27t\\_repeat\\_yourself](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself)).

**License** GPL (>= 3)

**URL** <https://github.com/kapsner/kdry>

**BugReports** <https://github.com/kapsner/kdry/issues>

**Depends** R (>= 2.10)

**Imports** data.table, doParallel, foreach, Hmisc, magrittr, parallel, stats, utils

**Suggests** ggplot2, lintr, survival, testthat (>= 3.0.1)

**Config/testthat/edition** 3

**Config/testthat/parallel** false

**Date/Publication** 2023-04-18 14:10:08 UTC

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**NeedsCompilation** no

**Author** Lorenz A. Kapsner [cre, aut, cph]  
(<https://orcid.org/0000-0003-1866-860X>)

**Maintainer** Lorenz A. Kapsner <lorenz.kapsner@gmail.com>

**Repository** CRAN

## R topics documented:

dtr_matrix2df . . . . .	2
icolnames . . . . .	3
list.append . . . . .	3

list.update . . . . .	4
misc_argument_catcher . . . . .	5
misc_duplicated_by_names . . . . .	6
misc_recursive_copy . . . . .	6
misc_subset_options . . . . .	7
mlh_outsample_row_indices . . . . .	8
mlh_reshape . . . . .	9
mlh_subset . . . . .	9
pch_check_available_cores . . . . .	10
pch_clean_up . . . . .	11
pch_register_parallel . . . . .	11
plt_parallel_coordinates . . . . .	12
rep_mean_sd . . . . .	13
rep_median_ci . . . . .	14
rep_median_iqr . . . . .	16
rep_pval . . . . .	17
rep_sum_pct . . . . .	18
sts_normalize . . . . .	19

<b>Index</b>	<b>20</b>
--------------	-----------

---

dtr_matrix2df	<i>dtr_matrix2df</i>
---------------	----------------------

---

## Description

Data transformation: Converts a `matrix` to `data.table` and encodes categorical variables as `factor`.

## Usage

```
dtr_matrix2df(matrix, cat_vars = NULL)
```

## Arguments

<code>matrix</code>	An R <code>matrix</code> object.
<code>cat_vars</code>	A character vector with colnames that should be converted to <code>factor</code> (default: <code>NULL</code> ).

## Value

A `data.table` is returned.

**Examples**

```
data("iris")
mat <- data.matrix(iris)
dataset <- dtr_matrix2df(mat)
str(dataset)

dataset <- dtr_matrix2df(mat, cat_vars = "Species")
str(dataset)
```

---

icolnames	<i>icolnames</i>
-----------	------------------

---

**Description**

Return colnames in a table with index numbers.

**Usage**

```
icolnames(df)
```

**Arguments**

df                    A data.frame object.

**Value**

A data.table with the two columns index and name is returned.

**Examples**

```
data("iris")
icolnames(iris)
```

---

list.append	<i>list.append</i>
-------------	--------------------

---

**Description**

Helper function to append an R list.

**Usage**

```
list.append(main_list, append_list, ...)
```

**Arguments**

`main_list`      A list, to which another should be appended.  
`append_list`    A list to append to `main_list` (can be NULL or empty).  
`...`            Further arguments passed to `utils::modifyList()`.

**Details**

This function is a save wrapper around `utils::modifyLists` to combine lists as it checks for the input types and only appends the new list if its length is greater than 0.

**Value**

A list is returned.

**See Also**

[utils::modifyList\(\)](#)

**Examples**

```
l1 <- list("a" = 1, "b" = 2)
l2 <- list("c" = 3, "d" = 4)
list.append(l1, l2)
```

---

`list.update`

*list.update*

---

**Description**

Helper function to update items in an R list.

**Usage**

```
list.update(main_list, new_list, ...)
```

**Arguments**

`main_list`      A list, which items should be updated.  
`new_list`        A list with new values of items ffrom `main_list` that should be updated. All names of `new_list` must be present in `main_list`.  
`...`            Further arguments passed to `utils::modifyList()`.

**Details**

This function is a save wrapper around `utils::modifyLists` to update items in R lists as it checks for the input types and only accepts named lists.

**Value**

A list is returned.

**See Also**

`utils::modifyList()`

**Examples**

```
l1 <- list("a" = 1, "b" = 2)
l2 <- list("a" = 3, "b" = 4)
list.update(l1, l2)
```

---

`misc_argument_catcher misc_argument_catcher`

---

**Description**

Miscellaneous helper function to type-save catch arguments passed with R's ellipsis ("...").

**Usage**

```
misc_argument_catcher(...)
```

**Arguments**

...                   Named arguments passed to a function.

**Details**

This function aims at catching arguments that have been passed to an R function using R's ellipsis ("..."). Its purpose is to catch these arguments even in the case, if a list with arguments was provided to the ellipsis.

**Value**

A list is returned.

**Examples**

```
misc_argument_catcher(a = 1)
misc_argument_catcher(a = 1, b = 2, c = 3, d = "car")
misc_argument_catcher(list(a = 1, b = 2, c = 3, d = "car"))
misc_argument_catcher(list(a = 1, b = 2, c = 3, d = "car"), f = 9)
```

misc\_duplicated\_by\_names  
*misc\_duplicated\_by\_names*

---

**Description**

Miscellaneous helper function to detect items in an object with duplicated names, e.g. in named vectors or named lists.

**Usage**

```
misc_duplicated_by_names(object, ...)
```

**Arguments**

object	An R object that has names.
...	Named arguments passed on to duplicated.

**Value**

Returns a logical vector of length(object) with TRUE indicating the identified items with duplicated names.

**See Also**

[base::duplicated\(\)](#)

**Examples**

```
misc_duplicated_by_names(list(a = 1, a = 1))
```

---

misc\_recursive\_copy    *misc\_recursive\_copy*

---

**Description**

Recursively copying directories and subdirectories.

**Usage**

```
misc_recursive_copy(source_dir, target_dir, force = FALSE)
```

**Arguments**

source_dir	A character string. The path to the directory to be copied.
target_dir	A character string. The target path.
force	A boolean. If FALSE (the default) a dialog is prompted to ask the user if the directories should be copied recursively. If TRUE, the dialog is not prompted and the source_dir is copied directly to target_dir.

**Value**

This function has no return value.

**Examples**

```
if (interactive()) {  
  d1 <- file.path(tempdir(), "folder1")  
  d2 <- file.path(d1, "folder2")  
  d3 <- file.path(tempdir(), "new_folder")  
  f1 <- file.path(d1, "file.one")  
  dir.create(d2, recursive = TRUE)  
  file.create(f1)  
  misc_recursive_copy(d1, d3)  
}
```

---

misc\_subset\_options    *misc\_subset\_options*

---

**Description**

Miscellaneous helper function to subset R options by a keyword.

**Usage**

```
misc_subset_options(keyword)
```

**Arguments**

keyword	A character. The keyword to subset the R options.
---------	---

**Details**

This function subsets R's options() by a keyword. It returns a list of all available options that match with the keyword. The keyword is evaluated as a regular expression.

**Value**

A list is returned, containing the subset of R's options() that matches with the keyword.

**Examples**

```
misc_subset_options("default")
```

---

```
mlh_outsample_row_indices
      mlh_outsample_row_indices
```

---

**Description**

Machine learning helper function to convert a vector of (in- sample) row indices of a fold into out-of-sample row indices.

**Usage**

```
mlh_outsample_row_indices(fold_list, dataset_nrows, type = NULL)
```

**Arguments**

<code>fold_list</code>	A list of integer vectors that describe the row indices of cross-validation folds. The list must be named.
<code>dataset_nrows</code>	An integer. The number of rows in the dataset <code>dataset</code> . This parameter is required in order to compute the out-of-sample row indices.
<code>type</code>	A character. To be used if the out-of-sample row indices need to be formatted in a special manner (default: <code>NULL</code> ). Currently, the only allowed value is <code>type = "glmnet"</code> in order to format the row indices as required by <code>glmnet::cv.glmnet</code> 's argument <code>foldid</code> .

**Value**

If `type = NULL`, returns a list of same length as `fold_list` with each item containing a vector of out-of-sample row indices. If `type = "glmnet"`, a `data.table` is returned with two columns and each row representing one observation of the dataset that is assigned to a specific test fold. The column `"fold_id"` should be passed further on to the argument `foldid` of `glmnet::cv.glmnet`.

**Examples**

```
fold_list <- list(
  "Fold1" = setdiff(seq_len(100), 1:33),
  "Fold2" = setdiff(seq_len(100), 66:100),
  "Fold3" = setdiff(seq_len(100), 34:65)
)
mlh_outsample_row_indices(fold_list, 100)
mlh_outsample_row_indices(fold_list, 100, "glmnet")
```



---

mlh_reshape	<i>mlh_reshape</i>
-------------	--------------------

---

**Description**

Machine learning helper function to reshape a matrix of predicted probabilities to classes.

**Usage**

```
mlh_reshape(object)
```

**Arguments**

object	A matrix with predicted probabilities for several classes. Each row must sum up to 1.
--------	---

**Value**

Returns a vector of type factor of the same length as rows in object, representing the class with the highest probability for each observation in object.

**Examples**

```
set.seed(123)
class_0 <- rbeta(100, 2, 4)
class_1 <- (1 - class_0) * 0.4
class_2 <- (1 - class_0) * 0.6
dataset <- cbind("0" = class_0, "1" = class_1, "2" = class_2)
mlh_reshape(dataset)
```

---

mlh_subset	<i>mlh_subset</i>
------------	-------------------

---

**Description**

Machine learning helper function to select a subset from a data matrix or a response vector.

**Usage**

```
mlh_subset(object, ids)
```

**Arguments**

object	A vector or a data matrix. Supports also subsetting of "Surv" objects.
ids	An integer vector specifying the indices that should be selected from the object.

**Value**

Returns the specified subset of the object.

**Examples**

```
data("iris")
mlh_subset(iris, c(1:30))
mlh_subset(iris[, 5], c(1:30))
```

---

```
pch_check_available_cores
      pch_check_available_cores
```

---

**Description**

Parallel computing helper function to check for the available cores.

**Usage**

```
pch_check_available_cores(ncores = -1L)
```

**Arguments**

`ncores` An integer. A number of cores requested for parallel computing (default: -1L).

**Value**

The function returns an integer that indicates the number of cores available. If `ncores <= parallel::detectCores()` the function returns `ncores`. If `ncores > parallel::detectCores()`, the function returns `parallel::detectCores() - 1L`.

**Examples**

```
pch_check_available_cores(2)
```

---

pch\_clean\_up            *pch\_clean\_up*

---

**Description**

Parallel computing helper function to clean up the parallel backend.

**Usage**

```
pch_clean_up(cl)
```

**Arguments**

`cl`                    A cluster object of class `c("SOCKcluster", "cluster")`.

**Value**

The function returns nothing. Internally, it calls `parallel::stopCluster()` and `foreach::registerDoSEQ()`.

**See Also**

[parallel::stopCluster\(\)](#), [foreach::registerDoSEQ\(\)](#)

**Examples**

```
if (require("doParallel") && require("foreach")) {
  cl <- pch_register_parallel(pch_check_available_cores(2))
  pch_clean_up(cl)
}
```

---

pch\_register\_parallel    *pch\_register\_parallel*

---

**Description**

Parallel computing helper function to register a parallel backend.

**Usage**

```
pch_register_parallel(ncores)
```

**Arguments**

`ncores`                An integer. A number of cores requested for parallel computing (default: `-1L`).

**Value**

The function returns a object of class c("SOCKcluster", "cluster"), created with `parallel::makePSOCKcluster()`.

**See Also**

`parallel::makePSOCKcluster()`, `doParallel::registerDoParallel()`

**Examples**

```
if (require("doParallel") && require("foreach")) {
  cl <- pch_register_parallel(pch_check_available_cores(2))
  pch_clean_up(cl)
}
```

---

`plt_parallel_coordinates`

*plt\_parallel\_coordinates*

---

**Description**

Parallel coordinates plot

**Usage**

```
plt_parallel_coordinates(
  data,
  cols = NULL,
  color_variable = NULL,
  color_args = list(alpha = 0.6, begin = 0.1, end = 0.9, option = "inferno", direction =
    1),
  line_jitter = list(w = 0.04, h = 0.04),
  text_label_size = 3.5
)
```

**Arguments**

<code>data</code>	A <code>data.table</code> object with the columns containing the parameters to be plotted with the parallel coordinates plot.
<code>cols</code>	A character vector with column names to subset data (default: <code>NULL</code> ).
<code>color_variable</code>	A character. The name of the column to be used to colorize the lines of the plot (default: <code>NULL</code> ).
<code>color_args</code>	A list with parameters for the color gradient (see details).
<code>line_jitter</code>	A list with the elements <code>w</code> and <code>h</code> to define a line jitter (default: <code>list(w = 0.04, h = 0.04)</code> ), which are passed further on to define the position of <code>ggplot2::geom_line()</code> .
<code>text_label_size</code>	A numeric value to define the size of the text annotations (default: 3.5).

## Details

The color gradient of the plotted lines can be defined with a list provided to the argument `color_args`. Its default values are `alpha = 0.6`, `begin = .1`, `end = .9`, `option = "inferno"`, and `direction = 1` and are passed further on to `ggplot2::scale_color_viridis_c()`. The implementation to display categorical variables is still experimental.

## Value

Returns a parallel coordinates plot as `ggplot2` object.

## See Also

[ggplot2::scale\\_color\\_viridis\\_c\(\)](#)

## Examples

```
if (require("ggplot2")) {
  data("iris")
  plt_parallel_coordinates(
    data = data.table::as.data.table(iris[, -5]),
    cols = colnames(iris)[c(-1, -5)],
    color_variable = "Sepal.Length"
  )
}
```

---

rep\_mean\_sd

*rep\_mean\_sd*

---

## Description

Reporting helper function: computes and formats mean and standard deviation from a numeric vector.

## Usage

```
rep_mean_sd(
  x,
  digits = 2,
  na.rm = TRUE,
  sd_brackets = c("round", "square"),
  sd_prefix = TRUE,
  weighted = FALSE,
  weights = NA
)
```

**Arguments**

<code>x</code>	A numeric vector.
<code>digits</code>	An integer indicating the number of decimal places.
<code>na.rm</code>	A logical indicating if missings should be removed from <code>x</code> before computing the distributional parameters (default: TRUE).
<code>sd_brackets</code>	A character. Either "round" (default) or "square" to indicate the type of brackets to surround the standard deviation in the formatted output.
<code>sd_prefix</code>	A logical. If TRUE (default), the standard deviation is prefixed with a plus-minus sign ("±").
<code>weighted</code>	A logical. If TRUE, a weighted mean and standard deviation are calculated (default: FALSE).
<code>weights</code>	A vector with the weights (if <code>weighted = TRUE</code> ; default: NA) passed further on to <code>stats::weighted.mean()</code> and <code>Hmisc::wtd.var()</code> .

**Value**

A character with the formatted output.

**See Also**

`mean()`, `stats::sd()`, `stats::weighted.mean()`, `Hmisc::wtd.var()`

**Examples**

```
set.seed(123)
x <- rnorm(1000)
rep_mean_sd(x)
rep_mean_sd(rep(1, 10))
rep_mean_sd(x, sd_brackets = "square")
rep_mean_sd(x, sd_brackets = "square", sd_prefix = FALSE)
```

---

rep\_median\_ci

*rep\_median\_ci*

---

**Description**

Reporting helper function: computes and formats median and confidence interval from a numeric vector.

**Usage**

```
rep_median_ci(
  x,
  conf_int,
  digits = 2,
  na.rm = TRUE,
  collapse = "to",
  iqr_brackets = c("round", "square"),
  iqr_prefix = TRUE,
  weighted = FALSE,
  weights = NA
)
```

**Arguments**

x	A numeric vector.
conf_int	A numeric between 0 and 100 to indicate the confidence interval that should be computed.
digits	An integer indicating the number of decimal places.
na.rm	A logical indicating if missings should be removed from x before computing the distributional parameters (default: TRUE).
collapse	A character which is placed between the lower and the upper confidence bound in the formatted output.
iqr_brackets	A character. Either "round" (default) or "square" to indicate the type of brackets to surround the confidence interval in the formatted output.
iqr_prefix	A logical. If TRUE (default), the confidence interval is prefixed with "IQR: ".
weighted	A logical. If TRUE, a weighted median and confidence interval are calculated (default: FALSE).
weights	A numeric vector of weights passed further on to <code>Hmisc::wtd.quantile()</code> if <code>weighted = TRUE</code> (default: NA).

**Value**

A character with the formatted output.

**See Also**

[stats::median](#), [stats::quantile](#), [Hmisc::wtd.quantile\(\)](#)

**Examples**

```
set.seed(123)
x <- rnorm(1000)
rep_median_ci(x, conf_int = 95)
rep_median_ci(rep(1, 10), conf_int = 95)
rep_median_ci(x, conf_int = 95, collapse = "-")
rep_median_ci(x, iqr_brackets = "square", conf_int = 50)
```

---

rep_median_iqr	<i>rep_median_iqr</i>
----------------	-----------------------

---

### Description

Reporting helper function: computes and formats median and interquartile range from a numeric vector.

### Usage

```
rep_median_iqr(  
  x,  
  digits = 2,  
  na.rm = TRUE,  
  collapse = "to",  
  iqr_brackets = c("round", "square"),  
  iqr_prefix = TRUE  
)
```

### Arguments

x	A numeric vector.
digits	An integer indicating the number of decimal places.
na.rm	A logical indicating if missings should be removed from x before computing the distributional parameters (default: TRUE).
collapse	A character which is placed between the lower and the upper confidence bound in the formatted output.
iqr_brackets	A character. Either "round" (default) or "square" to indicate the type of brackets to surround the confidence interval in the formatted output.
iqr_prefix	A logical. If TRUE (default), the confidence interval is prefixed with "IQR: ".

### Details

This is just a special case of [rep\\_median\\_ci\(\)](#) with the parameter `conf_int` set to 50.

### Value

A character with the formatted output.

### See Also

[rep\\_median\\_ci\(\)](#)



**Examples**

```
set.seed(123)
x <- rnorm(1000)
rep_median_iqr(x)
rep_median_iqr(rep(1, 10))
rep_median_iqr(x, collapse = "-")
rep_median_iqr(x, iqr_brackets = "square")
rep_median_iqr(x, iqr_brackets = "square", iqr_prefix = FALSE)
rep_median_iqr(x, collapse = ";", iqr_prefix = FALSE)
```

---

rep\_pval

*rep\_pval*

---

**Description**

Reporting helper function: formats p-value.

**Usage**

```
rep_pval(p, threshold = 0.001, digits = 3L)
```

**Arguments**

<code>p</code>	The p-value that should be formatted.
<code>threshold</code>	A threshold to indicate that only "< threshold" is printed as output (default: 0.001).
<code>digits</code>	The number of digits of the formatted p-value (digits).

**Details**

If the p-value is lower than the threshold, the output of the function is "< threshold". Otherwise, the p-value is formatted to the number of digits.

**Value**

A character with the formatted p-value.

**Examples**

```
rep_pval(0.032)
rep_pval(0.00032)
```

---

rep_sum_pct	<i>rep_sum_pct</i>
-------------	--------------------

---

### Description

Reporting helper function: computes and formats the relative percentage of a count.

### Usage

```
rep_sum_pct(
  count,
  count_reference,
  digits = 2,
  na.rm = TRUE,
  brackets = c("round", "square"),
  suffix = TRUE
)
```

### Arguments

count	A numeric. The numerator.
count_reference	A numeric. The denominator.
digits	An integer indicating the number of decimal places.
na.rm	A logical indicating if missings should be removed from x before computing the distributional parameters (default: TRUE).
brackets	A character. Either "round" (default) or "square" to indicate the type of brackets to surround the relative count.
suffix	A character which is placed between the lower and the upper confidence bound in the formatted output.

### Value

A character with the formatted output.

### See Also

[stats::median](#), [stats::quantile](#), [Hmisc:wtd.quantile\(\)](#)

### Examples

```
rep_sum_pct(count = 40, count_reference = 200)
rep_sum_pct(count = 40, count_reference = 200, brackets = "square")
rep_sum_pct(40, 200, brackets = "square", suffix = FALSE)
```

---

sts_normalize	<i>sts_normalize</i>
---------------	----------------------

---

**Description**

Statistic helper function to normalize a continuous variable between zero and one.

**Usage**

```
sts_normalize(x, na.rm = FALSE)
```

**Arguments**

x	A vector of type numeric.
na.rm	A logical to indicate, if missings should be removed.

**Value**

Returns a vector of same length as x with values normalized between zero and one. If x contains missings and na.rm = TRUE, the missings are removed before normalization; otherwise, a vector of NA is returned.

**Examples**

```
sts_normalize(1:100)
```

# Index

`base::duplicated()`, 6

`doParallel::registerDoParallel()`, 12

`dtr_matrix2df`, 2

`foreach::registerDoSEQ()`, 11

`ggplot2::scale_color_viridis_c()`, 13

`Hmisc::wtd.quantile()`, 15, 18

`Hmisc::wtd.var()`, 14

`icolnames`, 3

`list.append`, 3

`list.update`, 4

`mean()`, 14

`misc_argument_catcher`, 5

`misc_duplicated_by_names`, 6

`misc_recursive_copy`, 6

`misc_subset_options`, 7

`mlh_outsample_row_indices`, 8

`mlh_reshape`, 9

`mlh_subset`, 9

`parallel::makePSOCKcluster()`, 12

`parallel::stopCluster()`, 11

`pch_check_available_cores`, 10

`pch_clean_up`, 11

`pch_register_parallel`, 11

`plt_parallel_coordinates`, 12

`rep_mean_sd`, 13

`rep_median_ci`, 14

`rep_median_ci()`, 16

`rep_median_iqr`, 16

`rep_pval`, 17

`rep_sum_pct`, 18

`stats::median`, 15, 18

`stats::quantile`, 15, 18

`stats::sd()`, 14

`stats::weighted.mean()`, 14

`sts_normalize`, 19

`utils::modifyList()`, 4, 5