# Package ‘kdry’

March 8, 2024

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

**Version** 0.0.2

**Description** A collection of personal helper functions 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>).

**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** 2024-03-08 13:30:02 UTC

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**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` *dtr_matrix2df*

---

## Description

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

## Usage

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

## Arguments

| | |
|---|---|
| `matrix` | An R `matrix` object. |
| `cat_vars` | A character vector with colnames that should be converted to `factor` (default: NULL). |

## 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 *icolnames*

---

### 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 *list.append*

---

### 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 from 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 purpos 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

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

## 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` *mlh_reshape*

## 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` *mlh_subset*

## 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

| | |
|---|---|
| `data` | A data.table object with the columns containing the parameters to be plotted with the parallel coordinates plot. |
| `cols` | A character vector with column names to subset `data` (default: `NULL`). |
| `color_variable` | A character. The name of the column to be used to colorize the lines of the plot (default: `NULL`). |
| `color_args` | A list with parameters for the color gradient (see details). |
| `line_jitter` | A list with the elements `w` and `h` to define a line jitter (default: `list(w = 0.04, h = 0.04)`), which are passed further on to define the position of `ggplot2::geom_line()`. |
| `text_label_size` | 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 furhter 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_frac_pct *rep_frac_pct*

---

## Description

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

## Usage

```
rep_frac_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_frac_pct(count = 40, count_reference = 200)
rep_frac_pct(count = 40, count_reference = 200, brackets = "square")
rep_frac_pct(40, 200, brackets = "square", suffix = FALSE)
```

---

## 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

| | |
|---|---|
| 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). |
| sd_brackets | A character. Either "round" (default) or "square" to indicate the type of brackets to surround the standard deviation in the formatted output. |
| sd_prefix | A logical. If TRUE (default), the standard deviation is prefixed with a plus-minus-sign ("±"). |
| weighted | A logical. If TRUE, a weighted mean and standard deviation are calculated (default: FALSE). |
| weights | A vector with the weights (if weighted = TRUE; default: NA) passed futher on to stats::weighted.mean() and Hmisc::wtd.var(). |

### 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 Hmisc::wtd.quantile() if weighted = TRUE (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 *rep_median_iqr*

## 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

| | |
|---|---|
| `p` | The p-value that should be formatted. |
| `threshold` | A threshold to indicate that only "< threshold" is printed as output (default: 0.001). |
| `digits` | 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 *rep_sum_pct*

## 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 *sts_normalize*

## 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 returend.

## Examples

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

# Index