

# Package ‘dynutils’

February 21, 2020

**Type** Package

**Title** Common Functionality for the 'dynverse' Packages

**Version** 1.0.5

**Description** Provides common functionality for the 'dynverse' packages.  
'dynverse' is created to support the development, execution, and benchmarking of trajectory inference methods.  
For more information, check out <<https://dynverse.org>>.

**License** MIT + file LICENSE

**URL** <https://github.com/dynverse/dynutils>

**BugReports** <https://github.com/dynverse/dynutils/issues>

**LazyData** TRUE

**RoxygenNote** 7.0.2

**Depends** R (>= 3.0.0)

**Imports** assertthat, crayon, desc, dplyr, magrittr, Matrix, methods,  
proxyC, purrr, Rcpp, remotes, stringr, tibble

**Suggests** ggplot2, hdf5r, knitr, readr, rmarkdown, testthat

**LinkingTo** Rcpp

**Encoding** UTF-8

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** Robrecht Cannoodt [aut, cre] (<<https://orcid.org/0000-0003-3641-729X>>,  
rcannood),  
Wouter Saelens [aut] (<<https://orcid.org/0000-0002-7114-6248>>, zouter)

**Maintainer** Robrecht Cannoodt <[rcannood@gmail.com](mailto:rcannood@gmail.com)>

**Repository** CRAN

**Date/Publication** 2020-02-21 12:30:02 UTC

**R topics documented:**

add_class	2
all_in	3
apply_minmax_scale	4
apply_quantile_scale	4
apply_uniform_scale	5
calculate_distance	5
calculate_mean	6
check_packages	7
dynutils	7
euclidean_distance	9
expand_matrix	9
extend_with	10
extract_row_to_list	10
has_names	11
inherit_default_params	12
install_packages	13
is_bounded	13
is_single_numeric	14
is_sparse	15
list_as_tibble	15
mapdf	16
project_to_segments	18
random_time_string	19
read_h5	19
recent_news	20
safe_tempdir	20
scale_minmax	21
scale_quantile	21
scale_uniform	22
switch_devel	23
test_h5_installation	23
tibble_as_list	24
<b>Index</b>	<b>25</b>

---

 add\_class

*Add class to object whilst keeping the old classes*


---

**Description**

Add class to object whilst keeping the old classes

**Usage**

```
add_class(x, class)
```

**Arguments**

x	a R object
class	A character vector naming classes

**Examples**

```
library(purrr)
l <- list(important_number = 42) %>% add_class("my_list")
```

---

all_in	<i>Check whether a vector are all elements of another vector</i>
--------	--

---

**Description**

Check whether a vector are all elements of another vector

**Usage**

```
all_in(x, table)

x %all_in% table
```

**Arguments**

x	The values to be matched.
table	The values to be matched against.

**Examples**

```
## Not run:
library(assertthat)
assert_that(c(1, 2) %all_in% c(0, 1, 2, 3, 4))
# TRUE

assert_that("a" %all_in% letters)
# TRUE

assert_that("A" %all_in% letters)
# Error: "A" is missing 1 element from letters: "A"

assert_that(1:10 %all_in% letters)
# Error: 1:10 is missing 10 elements from letters: 1L, 2L, 3L, ...

## End(Not run)
```

---

apply\_minmax\_scale     *Apply a minmax scale.*

---

**Description**

Anything outside the range of [0, 1] will be set to 0 or 1.

**Usage**

```
apply_minmax_scale(x, addend, multiplier)
```

**Arguments**

x	A numeric vector, matrix or data frame.
addend	A minimum vector for each column
multiplier	A scaling vector for each column

**Value**

The scaled matrix or vector. The numeric centering and scalings used are returned as attributes.

---

apply\_quantile\_scale     *Apply a quantile scale.*

---

**Description**

Anything outside the range of [0, 1] will be set to 0 or 1.

**Usage**

```
apply_quantile_scale(x, addend, multiplier)
```

**Arguments**

x	A numeric vector, matrix or data frame.
addend	A minimum vector for each column
multiplier	A scaling vector for each column

**Value**

The scaled matrix or vector. The numeric centering and scalings used are returned as attributes.

---

apply\_uniform\_scale     *Apply a uniform scale*

---

**Description**

Apply a uniform scale

**Usage**

```
apply_uniform_scale(x, addend, multiplier)
```

**Arguments**

x	A numeric vector, matrix or data frame.
addend	A centering vector for each column
multiplier	A scaling vector for each column

**Value**

The centered, scaled matrix. The numeric centering and scalings used are returned as attributes.

---

calculate\_distance     *Calculate (column-wise) distances/similarity between two matrices*

---

**Description**

These matrices can be dense or sparse.

**Usage**

```
calculate_distance(
  x,
  y = NULL,
  method = c("pearson", "spearman", "cosine", "euclidean", "manhattan"),
  margin = 1
)

list_distance_methods()

calculate_similarity(
  x,
  y = NULL,
  margin = 1,
  method = c("spearman", "pearson", "cosine")
)

list_similarity_methods()
```

**Arguments**

x	A numeric matrix, dense or sparse.
y	(Optional) a numeric matrix, dense or sparse, with <code>nrow(x) == nrow(y)</code> .
method	Which distance method to use. Options are: "cosine", "pearson", "spearman", "euclidean", and "manhattan".
margin	Which margin to use for the pairwise comparison. 1 => rowwise, 2 => columnwise.

**Examples**

```
## Generate two matrices with 50 and 100 samples
library(Matrix)
x <- Matrix::rsparsematrix(50, 1000, .01)
y <- Matrix::rsparsematrix(100, 1000, .01)

dist_euclidean <- calculate_distance(x, y, method = "euclidean")
dist_manhattan <- calculate_distance(x, y, method = "manhattan")
dist_spearman <- calculate_distance(x, y, method = "spearman")
dist_pearson <- calculate_distance(x, y, method = "pearson")
dist_angular <- calculate_distance(x, y, method = "cosine")
```

---

calculate_mean	<i>Calculate a (weighted) mean between vectors or a list of vectors</i>
----------------	---

---

**Description**

This function supports the arithmetic, geometric and harmonic mean.

**Usage**

```
calculate_mean(..., method, weights = NULL)

calculate_harmonic_mean(..., weights = NULL)

calculate_geometric_mean(..., weights = NULL)

calculate_arithmetic_mean(..., weights = NULL)
```

**Arguments**

...	Can be: <ul style="list-style-type: none"> <li>• One numeric vector</li> <li>• A list containing numeric vectors</li> <li>• Numeric vectors given as separate inputs</li> </ul>
method	The aggregation function. Must be one of "arithmetic", "geometric", and "harmonic".
weights	Weights with the same length as ....

**Examples**

```
calculate_arithmetic_mean(0.1, 0.5, 0.9)
calculate_geometric_mean(0.1, 0.5, 0.9)
calculate_harmonic_mean(0.1, 0.5, 0.9)
calculate_mean(.1, .5, .9, method = "harmonic")

# example with multiple vectors
calculate_arithmetic_mean(c(0.1, 0.9), c(0.2, 1))

# example with a list of vectors
vectors <- list(c(0.1, 0.2), c(0.4, 0.5))
calculate_geometric_mean(vectors)

# example of weighted means
calculate_geometric_mean(c(0.1, 10), c(0.9, 20), c(0.5, 2), weights = c(1, 2, 5))
```

---

check_packages	<i>Check which packages are installed</i>
----------------	---

---

**Description**

Check which packages are installed

**Usage**

```
check_packages(...)
```

**Arguments**

... A set of package names

**Examples**

```
check_packages("SCORPIUS", "dynutils")
check_packages(c("princurve", "mlr", "tidyverse"))
```

---

dynutils	<i>Common functionality for the dynverse packages</i>
----------	---

---

**Description**

Provides common functionality for the dynverse packages. dynverse is created to support the development, execution, and benchmarking of trajectory inference methods. For more information, check out [dynverse.org](https://dynverse.org).

### Manipulation of lists

- `add_class()`: Add a class to an object
- `extend_with()`: Extend list with more data

### Calculations

- `calculate_distance()`: Calculate pairwise distances between two (sparse) matrices
- `calculate_similarity()`: Calculate pairwise similarities between two (sparse) matrices
- `calculate_mean()`: Calculate a (weighted) mean between vectors or a list of vectors; supports the arithmetic, geometric and harmonic mean
- `project_to_segments()`: Project a set of points to to set of segments

### Manipulation of matrices

- `expand_matrix()`: Add rows and columns to a matrix

### Scaling of matrices and vectors

- `scale_uniform()`: Rescale data to have a certain center and max range
- `scale_minmax()`: Rescale data to a [0, 1] range
- `scale_quantile()`: Cut off outer quantiles and rescale to a [0, 1] range

### Manipulation of functions

- `inherit_default_params()`: Have one function inherit the default parameters from other functions

### Manipulation of packages

- `check_packages()`: Easily checking whether certain packages are installed
- `install_packages()`: Install packages taking into account the remotes of another

### Manipulation of vectors

- `random_time_string()`: Generates a string very likely to be unique

### Tibble helpers

- `list_as_tibble()`: Convert a list of lists to a tibble whilst retaining class information
- `tibble_as_list()`: Convert a tibble back to a list of lists whilst retaining class information
- `extract_row_to_list()`: Extracts one row from a tibble and converts it to a list
- `mapdf()`: Apply a function to each row of a data frame

### File helpers

- `safe_tempdir()`: Create an empty temporary directory and return its path



**Assertion helpers**

- `%all_in%()`: Check whether a vector are all elements of another vector
- `%has_names%()`: Check whether an object has certain names
- `is_single_numeric()`: Check whether a value is a single numeric
- `is_bounded()`: Check whether a value within a certain interval

**Package helpers**

- `recent_news()`: Print the most recent news (assumes NEWS.md file as specified by `news()`)

---

euclidean\_distance      *These functions will be removed soon*

---

**Description**

Use `calculate_distance()` instead.

**Usage**

```
euclidean_distance(x, y = NULL)
```

```
correlation_distance(x, y = NULL)
```

**Arguments**

<code>x</code>	A numeric matrix, dense or sparse.
<code>y</code>	(Optional) a numeric matrix, dense or sparse, with <code>nrow(x) == nrow(y)</code> .

---

expand\_matrix      *Expand a matrix with given rownames and colnames*

---

**Description**

Expand a matrix with given rownames and colnames

**Usage**

```
expand_matrix(mat, rownames = NULL, colnames = NULL, fill = 0)
```

**Arguments**

<code>mat</code>	The matrix to expand
<code>rownames</code>	The desired rownames
<code>colnames</code>	The desired colnames
<code>fill</code>	With what to fill missing data

**Examples**

```
x <- matrix(runif(12), ncol = 4, dimnames = list(c("a", "c", "d"), c("D", "F", "H", "I")))
expand_matrix(x, letters[1:5], LETTERS[1:10], fill = 0)
```

---

extend_with	<i>Extend an object</i>
-------------	-------------------------

---

**Description**

Extend an object

**Usage**

```
extend_with(object, .class_name, ...)
```

**Arguments**

object	A list
.class_name	A class name to add
...	Extra information in the list

**Examples**

```
library(purrr)
l <- list(important_number = 42) %>% add_class("my_list")
l %>% extend_with(
  .class_name = "improved_list",
  url = "https://github.com/dynverse/dynverse"
)
l
```

---

extract_row_to_list	<i>Extracts one row from a tibble and converts it to a list</i>
---------------------	---

---

**Description**

Extracts one row from a tibble and converts it to a list

**Usage**

```
extract_row_to_list(tib, row_id)
```

**Arguments**

tib	the tibble
row_id	the index of the row to be selected, or alternatively an expression which will be evaluated to such an index

**Value**

the corresponding row from the tibble as a list

**See Also**

list\_as\_tibble tibble\_as\_list mapdf

**Examples**

```
library(tibble)

tib <- tibble(
  a = c(1, 2),
  b = list(log10, sqrt),
  c = c("parrot", "quest"),
  .object_class = list(c("myobject", "list"), c("yourobject", "list"))
)

extract_row_to_list(tib, 2)
extract_row_to_list(tib, which(a == 1))
```

---

has\_names

*Check whether an object has certain names*

---

**Description**

Check whether an object has certain names

**Usage**

```
has_names(x, which)
```

```
x %has_names% which
```

**Arguments**

x                    object to test

which                name

**Examples**

```
## Not run:
library(assertthat)
li <- list(a = 1, b = 2)

assert_that(li %has_names% "a")
# TRUE
```

```
assert_that(li %has_names% "c")
# Error: li is missing 1 name from "c": "c"

assert_that(li %has_names% letters)
# Error: li is missing 24 names from letters: "c", "d", "e", ...

## End(Not run)
```

---

inherit\_default\_params

*Inherit default parameters from a list of super functions*

---

## Description

Inherit default parameters from a list of super functions

## Usage

```
inherit_default_params(super_functions, fun)
```

## Arguments

super_functions	A list of super functions of which ‘fun’ needs to inherit the default parameters
fun	The function whose default parameters need to be overridden

## Value

Function fun, but with the default parameters of the super\_functions

## Examples

```
fun1 <- function(a = 10, b = 7) runif(a, -b, b)
fun2 <- function(c = 9) 2^c

fun3 <- inherit_default_params(
  super = list(fun1, fun2),
  fun = function(a, b, c) {
    list(x = fun1(a, b), y = fun2(c))
  }
)

fun3
```

---

install_packages	<i>Check package availability</i>
------------------	-----------------------------------

---

**Description**

If the session is interactive, prompt the user whether to install the packages.

**Usage**

```
install_packages(..., try_install = interactive())
```

**Arguments**

...	The names of the packages to be checked
try_install	Whether running interactively, which will prompt the user before installation

**Examples**

```
## Not run:  
install_packages("SCORPIUS")  
  
## End(Not run)
```

---

is_bounded	<i>Check whether a value within a certain interval</i>
------------	--

---

**Description**

Check whether a value within a certain interval

**Usage**

```
is_bounded(  
  x,  
  lower_bound = -Inf,  
  lower_closed = FALSE,  
  upper_bound = Inf,  
  upper_closed = FALSE  
)
```

**Arguments**

x	A value to be tested
lower_bound	The lower bound
lower_closed	Whether the lower bound is closed
upper_bound	The upper bound
upper_closed	Whether the upper bound is closed

**Examples**

```
## Not run:
library(assertthat)
assert_that(is_bounded(10))
# TRUE

assert_that(is_bounded(10:30))
# TRUE

assert_that(is_bounded(Inf))
# Error: Inf is not bounded by (-Inf,Inf)

assert_that(is_bounded(10, lower_bound = 20))
# Error: 10 is not bounded by (20,Inf)

assert_that(is_bounded(
  10,
  lower_bound = 20,
  lower_closed = TRUE,
  upper_bound = 30,
  upper_closed = FALSE
))
# Error: 10 is not bounded by [20,30)

## End(Not run)
```

---

is_single_numeric	<i>Check whether a value is a single numeric</i>
-------------------	--

---

**Description**

Check whether a value is a single numeric

**Usage**

```
is_single_numeric(x)
```

**Arguments**

x	A value to be tested
---	----------------------

**Examples**

```
## Not run:
library(assertthat)
assert_that(is_single_numeric(1))
# TRUE

assert_that(is_single_numeric(Inf))
```

```
# TRUE

assert_that(is_single_numeric(1.6))
# TRUE

assert_that(is_single_numeric(NA))
# Error: NA is not a single numeric value

assert_that(is_single_numeric(1:6))
# Error: 1:6 is not a single numeric value

assert_that(is_single_numeric("pie"))
# Error: "pie" is not a single numeric value

## End(Not run)
```

---

is_sparse	<i>Check if an object is a sparse matrix</i>
-----------	--

---

### Description

Check if an object is a sparse matrix

### Usage

```
is_sparse(x)
```

### Arguments

x                    An object to test

### Examples

```
is_sparse(matrix(1:10)) # FALSE
is_sparse(Matrix::rsparsematrix(100, 200, .01)) # TRUE
```

---

list_as_tibble	<i>Convert a list of lists to a tibble</i>
----------------	--

---

### Description

Convert a list of lists to a tibble

### Usage

```
list_as_tibble(list_of_rows)
```

**Arguments**

`list_of_rows` The list to be converted to a tibble

**Value**

A tibble with the same number of rows as there were elements in `list_of_rows`

**See Also**

`tibble_as_list` `extract_row_to_list` `mapdf`

**Examples**

```
library(purrr)

li <- list(
  list(a = 1, b = log10, c = "parrot") %>% add_class("myobject"),
  list(a = 2, b = sqrt, c = "quest") %>% add_class("yourobject")
)

tib <- list_as_tibble(li)

tib
```

---

mapdf

*Apply a function to each row of a data frame*

---

**Description**

The `mapdf` functions transform their input by applying a function to each row of a data frame and returning a vector the same length as the input. These functions work a lot like `purrr`'s `map()` functions.

**Usage**

```
mapdf(.x, .f, ...)

mapdf_lgl(.x, .f, ...)

mapdf_chr(.x, .f, ...)

mapdf_int(.x, .f, ...)

mapdf_dbl(.x, .f, ...)

mapdf_dfr(.x, .f, ...)

mapdf_dfc(.x, .f, ...)
```



```
mapdf_lat(.x, .f, ...)
```

```
walkdf(.x, .f, ...)
```

### Arguments

<code>.x</code>	A <code>data.frame</code> , <code>data_frame</code> , or <code>tibble</code> .
<code>.f</code>	A function or formula. If a function, the first argument will be the row as a list. If a formula, e.g. <code>~ .\$a</code> , the <code>.</code> is a placeholder for the row as a list.
<code>...</code>	Additional arguments passed on to the mapped function.

### Details

- `mapdf()` always returns a list.
- `mapdf_lgl()`, `mapdf_int()`, `mapdf_dbl()` and `mapdf_chr()` return vectors of the corresponding type (or die trying).
- `mapdf_dfr()` and `mapdf_dfc()` return data frames created by row-binding and column-binding respectively. They require `dplyr` to be installed.
- `mapdf_lat()` returns a tibble by transforming outputted lists to a tibble using [list\\_as\\_tibble](#).
- `walkdf()` calls `.f` for its side-effect and returns the input `.x`.

### Examples

```
library(dplyr)

tib <- tibble(
  a = c(1, 2),
  b = list(log10, sqrt),
  c = c("parrot", "quest"),
  .object_class = list(c("myobject", "list"), c("yourobject", "list"))
)

# map over the rows using a function
tib %>% mapdf(class)

# or use an anonymous function
tib %>% mapdf(function(row) paste0(row$b(row$a), "_", row$c))

# or a formula
tib %>% mapdf(~ .$b)

# there are many more variations available
# see ?mapdf for more info
tib %>% mapdf_lgl(~ .$a > 1)
tib %>% mapdf_chr(~ paste0("~", .$c, "~"))
tib %>% mapdf_int(~ nchar(.$c))
tib %>% mapdf_dbl(~ .$a * 1.234)
```

---

project\_to\_segments    *Project a set of points to to set of segments*

---

### Description

Finds the projection index for a matrix of points  $x$ , when projected onto a set of segments defined by `segment_start` and `segment_end`.

### Usage

```
project_to_segments(x, segment_start, segment_end)
```

### Arguments

`x`                    a matrix of data points.  
`segment_start`    a matrix of segment start points.  
`segment_end`       a matrix of segment end points.

### Value

A list with components

`x_proj`            a matrix of projections of  $x$  onto the given segments.  
`segment`           the index of the segment a point is projected on  
`progression`       the progression of a projection along its segment  
`distance`           the distance from each point in  $x$  to its projection in `x_proj`

### Examples

```
x <- matrix(rnorm(50, 0, .5), ncol = 2)
segfrom <- matrix(c(0, 1, 0, -1, 1, 0, -1, 0), ncol = 2, byrow = TRUE)
segto <- segfrom / 10
fit <- project_to_segments(x, segfrom, segto)

str(fit) # examine output
```

---

random_time_string	<i>Generate random string</i>
--------------------	-------------------------------

---

**Description**

Generate a random string with first the current time, together with a random number

**Usage**

```
random_time_string(name = NULL)
```

**Arguments**

name	Optional string to be added in the random_time_string
------	---

**Examples**

```
random_time_string("test")
```

---

read_h5	<i>Read/write R objects to a H5 file.</i>
---------	---

---

**Description**

Read/write R objects to a H5 file.

**Usage**

```
read_h5(path)
```

```
read_h5_(file_h5)
```

```
write_h5(x, path)
```

```
write_h5_(x, file_h5, path)
```

**Arguments**

path	Path to read from/write to.
file_h5	A H5 file to read from/write to.
x	R object to write.

---

recent_news	<i>Print the most recent news</i>
-------------	-----------------------------------

---

**Description**

Print the most recent news

**Usage**

```
recent_news(path = NULL, package = detect_package_name(path = path), n = 2)
```

**Arguments**

path	The path of the description in which the package resides
package	The package name
n	Number of recent news to print

---

safe_tmpdir	<i>Create an empty temporary directory and return its path</i>
-------------	--

---

**Description**

Create an empty temporary directory and return its path

**Usage**

```
safe_tmpdir(subfolder)
```

**Arguments**

subfolder	Name of a subfolder to be created
-----------	-----------------------------------

**Examples**

```
## Not run:  
safe_tmpdir("samson")  
# "/tmp/Rtmp8xCGJe/file339a13bec763/samson"  
  
## End(Not run)
```

---

scale_minmax	<i>Rescale data to a [0, 1] range</i>
--------------	---------------------------------------

---

**Description**

Rescale data to a [0, 1] range

**Usage**

```
scale_minmax(x)
```

**Arguments**

x                    A numeric vector, matrix or data frame.

**Value**

The centered, scaled matrix or vector. The numeric centering and scalings used are returned as attributes.

**Examples**

```
## Generate a matrix from a normal distribution
## with a large standard deviation, centered at c(5, 5)
x <- matrix(rnorm(200*2, sd = 10, mean = 5), ncol = 2)

## Minmax scale the data
x_scaled <- scale_minmax(x)

## Plot rescaled data
plot(x_scaled)

## Show ranges of each column
apply(x_scaled, 2, range)
```

---

scale_quantile	<i>Cut off outer quantiles and rescale to a [0, 1] range</i>
----------------	--

---

**Description**

Cut off outer quantiles and rescale to a [0, 1] range

**Usage**

```
scale_quantile(x, outlier_cutoff = 0.05)
```

**Arguments**

`x` A numeric vector, matrix or data frame.  
`outlier_cutoff` The quantile cutoff for outliers (default 0.05).

**Value**

The centered, scaled matrix or vector. The numeric centering and scalings used are returned as attributes.

**Examples**

```
## Generate a matrix from a normal distribution
## with a large standard deviation, centered at c(5, 5)
x <- matrix(rnorm(200*2, sd = 10, mean = 5), ncol = 2)

## Scale the dataset between [0,1]
x_scaled <- scale_quantile(x)

## Plot rescaled data
plot(x_scaled)

## Show ranges of each column
apply(x_scaled, 2, range)
```

---

scale_uniform	<i>Rescale data to have a certain center and max range.</i>
---------------	---

---

**Description**

`scale_uniform` uniformly scales a given matrix such that the returned space is centered on `center`, and each column was scaled equally such that the range of each column is at most `max_range`.

**Usage**

```
scale_uniform(x, center = 0, max_range = 1)
```

**Arguments**

`x` A numeric vector matrix or data frame.  
`center` The new center point of the data.  
`max_range` The maximum range of each column.

**Value**

The centered, scaled matrix. The numeric centering and scalings used are returned as attributes.

**Examples**

```
## Generate a matrix from a normal distribution
## with a large standard deviation, centered at c(5, 5)
x <- matrix(rnorm(200*2, sd = 10, mean = 5), ncol = 2)

## Center the dataset at c(0, 0) with a minimum of c(-.5, -.5) and a maximum of c(.5, .5)
x_scaled <- scale_uniform(x, center = 0, max_range = 1)

## Plot rescaled data
plot(x_scaled)

## Show ranges of each column
apply(x_scaled, 2, range)
```

---

switch_devel	<i>Switching of development stage within the dynverse</i>
--------------	---

---

**Description**

Switching of development stage within the dynverse

**Usage**

```
switch_devel(file = "DESCRIPTION", desc = desc::desc(file = file))
switch_master(file = "DESCRIPTION", desc = desc::desc(file = file))
switch_cran(file = "DESCRIPTION", desc = desc::desc(file = file))
```

**Arguments**

file	The description file, defaults to DESCRIPTION
desc	The read in description using the desc package

---

test_h5_installation	<i>Tests whether hdf5 is correctly installed and can load/write data</i>
----------------------	--

---

**Description**

Tests whether hdf5 is correctly installed and can load/write data

**Usage**

```
test_h5_installation(detailed = FALSE)

get_h5_test_data()
```

**Arguments**

detailed      Whether to do a detailed check

---

tibble\_as\_list      *Convert a tibble to a list of lists*

---

**Description**

Convert a tibble to a list of lists

**Usage**

```
tibble_as_list(tib)
```

**Arguments**

tib      A tibble

**Value**

A list with the same number of lists as there were rows in tib

**See Also**

list\_as\_tibble extract\_row\_to\_list mapdf

**Examples**

```
library(tibble)

tib <- tibble(
  a = c(1, 2),
  b = list(log10, sqrt),
  c = c("parrot", "quest"),
  .object_class = list(c("myobject", "list"), c("yourobject", "list"))
)

li <- tibble_as_list(tib)

li
```



# Index

`%all_in%` (`all_in`), 3  
`%has_names%` (`has_names`), 11  
`%all_in%`(), 9  
`%has_names%`(), 9

`add_class`, 2  
`add_class()`, 8  
`all_in`, 3  
`apply_minmax_scale`, 4  
`apply_quantile_scale`, 4  
`apply_uniform_scale`, 5

`calculate_arithmetic_mean`  
  (`calculate_mean`), 6  
`calculate_distance`, 5  
`calculate_distance()`, 8, 9  
`calculate_geometric_mean`  
  (`calculate_mean`), 6  
`calculate_harmonic_mean`  
  (`calculate_mean`), 6  
`calculate_mean`, 6  
`calculate_mean()`, 8  
`calculate_similarity`  
  (`calculate_distance`), 5  
`calculate_similarity()`, 8  
`check_packages`, 7  
`check_packages()`, 8  
`correlation_distance`  
  (`euclidean_distance`), 9

`dynutils`, 7

`euclidean_distance`, 9  
`expand_matrix`, 9  
`expand_matrix()`, 8  
`extend_with`, 10  
`extend_with()`, 8  
`extract_row_to_list`, 10  
`extract_row_to_list()`, 8

`get_h5_test_data`  
  (`test_h5_installation`), 23

`has_names`, 11

`inherit_default_params`, 12  
`inherit_default_params()`, 8  
`install_packages`, 13  
`install_packages()`, 8  
`is_bounded`, 13  
`is_bounded()`, 9  
`is_single_numeric`, 14  
`is_single_numeric()`, 9  
`is_sparse`, 15

`list_as_tibble`, 15, 17  
`list_as_tibble()`, 8  
`list_distance_methods`  
  (`calculate_distance`), 5  
`list_similarity_methods`  
  (`calculate_distance`), 5

`map()`, 16  
`mapdf`, 16  
`mapdf()`, 8  
`mapdf_chr` (`mapdf`), 16  
`mapdf_dbl` (`mapdf`), 16  
`mapdf_dfc` (`mapdf`), 16  
`mapdf_dfr` (`mapdf`), 16  
`mapdf_int` (`mapdf`), 16  
`mapdf_lat` (`mapdf`), 16  
`mapdf_lgl` (`mapdf`), 16

`news()`, 9

`project_to_segments`, 18  
`project_to_segments()`, 8

`random_time_string`, 19  
`random_time_string()`, 8  
`read_h5`, 19

`read_h5_(read_h5)`, 19  
`recent_news`, 20  
`recent_news()`, 9

`safe_tempdir`, 20  
`safe_tempdir()`, 8  
`scale_minmax`, 21  
`scale_minmax()`, 8  
`scale_quantile`, 21  
`scale_quantile()`, 8  
`scale_uniform`, 22  
`scale_uniform()`, 8  
`switch_cran (switch_devel)`, 23  
`switch_devel`, 23  
`switch_master (switch_devel)`, 23

`test_h5_installation`, 23  
`tibble_as_list`, 24  
`tibble_as_list()`, 8

`walkdf (mapdf)`, 16  
`write_h5 (read_h5)`, 19  
`write_h5_ (read_h5)`, 19