

# Package ‘itol.toolkit’

April 20, 2023

**Title** Helper Functions for 'Interactive Tree Of Life'

**Version** 1.1.5

**Description** The 'Interactive Tree Of Life' <<https://itol.embl.de/>> online server can edit and annotate trees interactively. The 'itol.toolkit' package can support all types of annotation templates.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.2.2

**Imports** dplyr, stringr, stats, seqinr, utils, methods, Biostrings, tidy, ape, data.table, purrr, wesanderson, miniUI, shiny, rstudioapi, colourpicker, ggsci, RColorBrewer

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**Depends** R (>= 2.10)

**LazyData** true

**Collate** 'addins.R' 'data.R' 'learn.R' 'object.R' 'utils.R' 'output.R' 'user.R' 'ops.R'

**NeedsCompilation** no

**Author** Tong Zhou [aut, cre]

**Maintainer** Tong Zhou <tongzhou2017@gmail.com>

**Repository** CRAN

**Date/Publication** 2023-04-20 18:52:30 UTC

## R topics documented:

+,itol.hub,itol.unit-method . . . . .	3
complex_html_text . . . . .	4
convert_01 . . . . .	4
convert_01_to_connect . . . . .	5

convert_range_to_node . . . . .	5
correct_get_color . . . . .	6
count_to_tree . . . . .	6
create_hub . . . . .	7
create_theme . . . . .	8
create_unit . . . . .	8
df_merge . . . . .	11
fa_read . . . . .	12
fa_write . . . . .	12
file_get_dir . . . . .	13
file_get_name . . . . .	13
file_to_unit . . . . .	14
get_color . . . . .	14
head_line . . . . .	15
hub_to_unit . . . . .	15
inbuilt_themes . . . . .	16
itol.hub-class . . . . .	17
itol.theme-class . . . . .	17
itol.unit-class . . . . .	18
learn_data . . . . .	18
learn_data_from_file . . . . .	19
learn_data_from_files . . . . .	19
learn_data_from_unit . . . . .	20
learn_data_from_unit_list . . . . .	20
learn_df . . . . .	21
learn_field . . . . .	21
learn_legend . . . . .	22
learn_line . . . . .	24
learn_profile . . . . .	25
learn_separator . . . . .	26
learn_subdf . . . . .	27
learn_theme_align . . . . .	27
learn_theme_alignment . . . . .	28
learn_theme_bar . . . . .	28
learn_theme_basic_plot . . . . .	29
learn_theme_basic_theme . . . . .	29
learn_theme_binary . . . . .	30
learn_theme_border . . . . .	30
learn_theme_common_themes . . . . .	31
learn_theme_connection . . . . .	31
learn_theme_domain . . . . .	32
learn_theme_externalshape . . . . .	32
learn_theme_heatmap . . . . .	33
learn_theme_image . . . . .	33
learn_theme_label . . . . .	34
learn_theme_linechart . . . . .	35
learn_theme_piechart . . . . .	36
learn_theme_specific_themes . . . . .	37

<code>learn_theme_strip_label</code> . . . . .	37
<code>learn_type</code> . . . . .	38
<code>line_clean</code> . . . . .	39
<code>line_split</code> . . . . .	40
<code>merge_unit</code> . . . . .	41
<code>show,itol.hub-method</code> . . . . .	41
<code>template_groups</code> . . . . .	42
<code>template_parameters_count</code> . . . . .	43
<code>unite_rows</code> . . . . .	43
<code>use.theme</code> . . . . .	44
<code>write_hub</code> . . . . .	45
<code>write_raw</code> . . . . .	46
<code>write_unit</code> . . . . .	47

**Index** **49**

---

`+,itol.hub,itol.unit-method`

*plus method add method for S4 class itol.hub and itol.unit*

---

**Description**

- plus method add method for S4 class itol.hub and itol.unit
- plus method add method for S4 class itol.unit and itol.unit

**Usage**

```
## S4 method for signature 'itol.hub,itol.unit'  
e1 + e2  
  
## S4 method for signature 'itol.unit,itol.unit'  
e1 + e2
```

**Arguments**

- `e1` An object of class itol.unit
- `e2` An object of class itol.unit

**Value**

- a itol.hub object with new data from itol.unit object
- a itol.unit object with merged data

---

complex_html_text	<i>Complex HTML text</i>
-------------------	--------------------------

---

**Description**

Interactively combine columns by HTML styles and record workflow as reproducible code.

**Usage**

```
complex_html_text()
```

**Details**

When you're done, the code performing this operation will be emitted at the cursor position.

---

convert_01	<i>Convert character data to 0/1</i>
------------	--------------------------------------

---

**Description**

In data frame and list, convert character and numeric data to 0/1.

**Usage**

```
convert_01(object)
```

**Arguments**

object	data frame or list
--------	--------------------

**Value**

a data frame with 0/1 values

---

convert\_01\_to\_connect *Convert 0/1 data to connection pairs*

---

**Description**

If two column has more than 1 shared element then they have connection. Convert 0/1 data to connection pairs in long shape table. The 0-connection pairs are removed.

**Usage**

```
convert_01_to_connect(object)
```

**Arguments**

object            data frame with 0/1 data

**Value**

a data frame with source and target connection information

---

convert\_range\_to\_node *Convert range to node id*

---

**Description**

Convert the data frame with range id to node id by mrca method.

**Usage**

```
convert_range_to_node(df, tree)
```

**Arguments**

df                data frame with any type of id  
tree              tree file path

**Value**

a data frame with converted id from range id

---

correct_get_color	<i>correct_get_color</i>
-------------------	--------------------------

---

**Description**

correct\_get\_color. (Version 0.0.0.9000)

**Usage**

```
correct_get_color(str)
```

**Arguments**

str	taxa string
-----	-------------

**Value**

a vector of colors

---

count_to_tree	<i>Calculate tree based on count matrix</i>
---------------	---

---

**Description**

While we start analysis from count matrix not sequences alignment, we could use clustering methods to get main tree in phylo object class of output as Newick format file. If the samples or elements have group information, we could use weighted clustering method to get a clear grouped structure.

**Usage**

```
count_to_tree(count, group = NULL, weight = 0)
```

**Arguments**

count	a data frame containing numeric values of abundance or other count.
group	a vector of character containing the group information. The length of the vector should be same with the count columns number. If using unweighted clustering, should ignore this parameter.
weight	a number specifying the weight size of the group information. In most case, 1 is enough. If the value is between 0 and 1, it will make the weight of group information weak. If the value is more than 1, it will make the weight of group information strong.

**Value**

a phylo class object containing

edge	a vector of integers specifying edge id. The length of vector is double of node number
edge.length	a vector of numbers specifying edge length
tip.label	a vector of character specifying the tip label
Nnode	a number specifying the number of nodes
node.label	a vector of character specifying the node label. If the tree calculated from count matrix or other case, the node label will generated by ape::makeNodeLabel function. And the Most Recent Common Ancestors(MRCA) node will be named with weighted group information, if the parameter group is not null.

---

create\_hub

*Create itol.hub Object*

---

**Description**

create a new object for itol.hub

**Usage**

```
create_hub(
  tree,
  field_tree = NULL,
  seq = NULL,
  abundance = NULL,
  taxonomy = NULL,
  node_data = NULL,
  tip_data = NULL
)
```

**Arguments**

tree	tree file
field_tree	todo
seq	todo
abundance	todo
taxonomy	todo
node_data	todo
tip_data	todo

**Value**

Returns a itol.hub object

**Examples**

```
TREE <- system.file("extdata", "tree_of_itol_templates.tree", package = "itol.toolkit")
create_hub(tree = TREE)
```

---

create\_theme                      *Create itol.theme Object*

---

**Description**

create a new object for itol.theme

**Usage**

```
create_theme(unit = NULL, file = NULL, tree = NULL, ...)
```

**Arguments**

unit	unit object
file	template file
tree	tree file
...	Further arguments to be passed to subsequent functions.

**Value**

Returns a itol.theme object

---

create\_unit                      *Create itol.unit*

---

**Description**

Create itol.unit from simple input in R environment.

**Usage**

```
create_unit(
  data,
  key,
  type,
  style = "default",
  subtype = NULL,
  color = NULL,
  line_type = NULL,
  font_type = NULL,
```



```

    size_factor = NULL,
    position = NULL,
    background_color = NULL,
    rotation = NULL,
    method = NULL,
    shape = NULL,
    fill = NULL,
    tree
)

```

### Arguments

data	if type == "COLLAPSE", a vector of characters specifying the tips or node used for collapsing used for extracting.
key	a character specifying the output file name for hub object.
type	a character specifying the template type used for extracting. Following choices are possible: "COLLAPSE", "PRUNE", "SPACING", "TREE_COLORS", "DATASET_STYLE", "LABEL", "DATASET_BINARY", "DATASET_GRADIENT", "DATASET_HEATMAP", "DATASET_SYMBOL", "DATASET_EXTERNALSHAPE", "DATASET_DOMAINS", "DATASET_SIMPLEBAR", "DATASET_MULTIBAR", "DATASET_BOXPLOT", "DATASET_LINECHART", "DATASET_PIECHART", "DATASET_ALIGNMENT", "DATASET_CONNECTION", "DATASET_IMAGE", "POPUP_INFO.
style	a character specifying the specific version of template type used for extracting. The default value is "default" style for all types.
subtype	a character specifying the subtype under type. If the type is "TREE_COLORS", the following choices are possible: "range", "clade", "branch", "label", "label_background".
color	a character specifying the color pattern name. The following choices are possible: "table2itol", "RColorBrewer", "ggsci".
line_type	a character specifying the normal or dashed line type used in clade and branch subtype.
font_type	a character specifying the bold, italic, and bold-italic font type used in label and branch subtype.
size_factor	a number specifying the line width used in clade and branch subtype and size factor in label subtype.
position	If type == "DATASET_STYLE", a character specifying the position: The following choices are possible: "node" and "clade". If type == "DATASET_TEXT", a number specifying the position of the text on the tree: -1 = external label; a number between 0 and 1 = internal label positioned at the specified value along the node branch (for example, position 0 is exactly at the start of node branch, position 0.5 is in the middle, and position 1 is at the end)
background_color	Only used while type == "DATASET_STYLE" and subtype == "label". a character or a vector of character specifying the background color in hexadecimal, RGB or RGBA notation.
rotation	Only used while type == "DATASET_TEXT". a number or a vector of number specifying the rotation angle of the text.

method	a character specifying the numeric data summarise method. If type == "DATASET_BINARY", the following choices are possible: "mean", "sum".
shape	a character or a vector of character specifying the symbol shape. If type == "DATASET_BINARY", the default is 2. If type == "DATASET_SYMBOL", the following choices are possible: 1 for rectangle, 2 for circle, 3 for star, 4 for left pointing triangle, 5 for right pointing triangle. If using NULL and there are data column, the functions will automatically help users to setup the shapes based on the levels of the data.
fill	If type == "DATASET_SYMBOL", 1/0 is specifying the shape outlier filled or not. If type == "DATASET_DOMAINS", the following choices are possible: "RE HH HV EL DI TR TL PL PR PU PD OC GP".
tree	a character specifying Newick format tree file path or a phylo object of main phylogenetic tree.

### Value

a itol.unit object containing

type	This group holds information about the template type of the data only. This is a very critical piece of information. In many functions of the itol.toolkit package, the template type information is used to determine the different data processing and input/output methods.
sep	This group holds data separator information only. This is one of the most important parameters for data reading and output. It is a separate category because it is frequently used and is an input parameter for other subsequent parameters to be read.
profile	This group contains basic information about the dataset, such as the dataset name and a color label to distinguish the dataset. The dataset name is extremely important. This parameter is used almost throughout the data processing of the itol.toolkit package. With the content of this parameter as the key value, the data and theme information of the dataset are associated. In turn, high throughput learning and writing of large-scale data can be achieved. This parameter is not included in some template types with a particularly simple structure, so we choose a file name or a user-defined method as the key value.
field	This group contains information about each sample within the dataset, and this type of parameter exists only for multi-sample data. This information even includes the clustering tree between samples. This information is usually stored as part of the column names in the metadata part or abundance information of the itol.hub object.
common_themes	These themes are used at high frequency in different templates. These parameters are small in number but constitute some common features of iTOL visual style settings, such as legend, margin, etc.
specific_themes	These themes are used only in specific templates. The number of these parameters is very large. However, most of them are used in only one template to control the style details of the visualization. By unifying these parameters and

calling them according to the template type, users can perform secondary development and data processing with a high degree of parameter aggregation without worrying too much about the differences between different template types.

**data** This slot contains a list of two data frames with the nodes and tips data separately. The first column of the two data frames is the node or tip id. If the input data contains range id, it would be converted to node id by the `convert_range_to_node` function automatically.

## Examples

```
tree <- system.file("extdata","tree_of_itol_templates.tree",package = "itol.toolkit")
data("template_groups")
data("template_parameters_count")
# COLLAPSE
group_names <- unique(template_groups$group)
object <- create_hub(tree = tree)
unit <- create_unit(data = group_names, key = "E001_collapse_1",
  type = "COLLAPSE", tree = tree)
object <- learn_data_from_unit(object,unit)
# PRUNE
select_note = c("theme_style","basic_plot")
unit <- create_unit(data = select_note, key = "E002_prune_1",
  type = "PRUNE", tree = tree)
object <- learn_data_from_unit(object,unit)
# SPACING
df_values = data.frame(id = row.names(template_parameters_count),
  values = rowSums(template_parameters_count))
unit <- create_unit(data = df_values, key = "E005_spacing_1",
  type = "SPACING", tree = tree)
object <- learn_data_from_unit(object,unit)
# TREE_COLORS
## range
unit <- create_unit(data = template_groups,
  key = "E006_tree_colors_1", type = "TREE_COLORS", subtype = "range",
  tree = tree)
object <- learn_data_from_unit(object,unit)
```

---

df\_merge

*Merge two data frame*

---

## Description

merge sub data frame into initial data frame

## Usage

```
df_merge(df1, df2, by = "id")
```

**Arguments**

df1	initial data frame
df2	sub data frame
by	key column

**Value**

a data frame containing merged information

---

fa_read	<i>Read fasta file</i>
---------	------------------------

---

**Description**

Read the fasta format sequences file into data.frame

**Usage**

```
fa_read(file)
```

**Arguments**

file	input file in fasta format
------	----------------------------

**Value**

a data frame with sequence id and sequence

---

fa_write	<i>Write fasta file</i>
----------	-------------------------

---

**Description**

Write the fasta format sequences file from data.frame. (Version 0.0.0.9000)

**Usage**

```
fa_write(object, file, id = "seq_name", seq = "sequence", append = FALSE)
```

**Arguments**

object	data.frame format data
file	input file in fasta format
id	id col
seq	seq col
append	append at the end of an already existing file

**Value**

No return value, only output a fasta file

---

file_get_dir	<i>Get file dir</i>
--------------	---------------------

---

**Description**

Get file dir from string

**Usage**

```
file_get_dir(str, up = FALSE)
```

**Arguments**

str	str
up	up dir

**Value**

a character specifying the dir path

---

file_get_name	<i>Get file name</i>
---------------	----------------------

---

**Description**

Get file name from string

**Usage**

```
file_get_name(str, with_ext = TRUE, keep_dir = FALSE)
```

**Arguments**

str	str
with_ext	with ext or not
keep_dir	keep file dir or not

**Value**

a character specifying the file name

---

file_to_unit	<i>Create itol.unit Object from file</i>
--------------	--

---

**Description**

create a new object for itol.unit

**Usage**

```
file_to_unit(file, tree, ...)
```

**Arguments**

file	template file
tree	tree file
...	Further arguments to be passed to subsequent functions.

**Value**

Returns a itol.unit object

---

get_color	<i>get_color</i>
-----------	------------------

---

**Description**

get color, support max length 40

**Usage**

```
get_color(n = 0, set = "table2itol")
```

**Arguments**

n	level length of a vector
set	a character specifying the palette set name. In default, table2itol is setted. The following choices are possible: wsanderson.

**Value**

a vector of colors

---

head_line	<i>head line</i>
-----------	------------------

---

**Description**

Head line for templates

**Usage**

head\_line(function\_name)

**Arguments**

function\_name    parent function name

**Value**

a character specifying the template type

---

hub_to_unit	<i>Create itol.unit Object from object</i>
-------------	--

---

**Description**

create a new object for itol.unit

**Usage**

hub\_to\_unit(object, theme, key)

**Arguments**

object	itol.hub object
theme	itol.theme object
key	key id of dataset name

**Value**

Returns a itol.unit object

---

`inbuilt_themes`*inbuilt themes*

---

**Description**

Default themes learned from iTOL official template examples.

**Usage**`inbuilt_themes`**Format**`inbuilt_themes:`

A list with 23 template themes:

**COLLAPSE** Default theme of collapse template

**PRUNE** Default theme of prune template

**SPACING** Default theme of spacing template

**TREE\_COLORS** Default theme of tree colors template

**DATASET\_STYLE** Default theme of style template

**LABELS** Default theme of labels template

**DATASET\_TEXT** Default theme of text template

**DATASET\_COLORSTRIP** Default theme of colorstrip template

**DATASET\_BINARY** Default theme of binary template

**DATASET\_GRADIENT** Default theme of gradient template

**DATASET\_HEATMAP** Default theme of heatmap template

**DATASET\_SYMBOL** Default theme of symbol template

**DATASET\_EXTERNALSHAPE** Default theme of externalshape template

**DATASET\_DOMAINS** Default theme of domains template

**DATASET\_SIMPLEBAR** Default theme of simple bar template

**DATASET\_MULTIBAR** Default theme of multi bar template

**DATASET\_BOXPLOT** Default theme of box plot template

**DATASET\_LINECHART** Default theme of line chart template

**DATASET\_PIECHART** Default theme of pie chart template

**DATASET\_ALIGNMENT** Default theme of alignment template

**DATASET\_CONNECTION** Default theme of connection template

**DATASET\_IMAGE** Default theme of image template

**POPUP\_INFO** Default theme of popup info template ...



---

itol.hub-class	<i>The itol.hub Class</i>
----------------	---------------------------

---

**Description**

The itol.hub object is an intermediate storage container used internally throughout the integration procedure to hold bits of data that are useful downstream.

**Slots**

tree a list of meta data table, usually raw, full, and analyze  
 seq identity of the active assay  
 abundance abundance  
 taxonomy taxonomy  
 meta.data other meta.data  
 theme itol theme

---

itol.theme-class	<i>The itol.theme Class</i>
------------------	-----------------------------

---

**Description**

The itol.theme object is an intermediate storage container used internally throughout the integration procedure to hold bits of data that are useful downstream.

**Slots**

type a list of meta data table, usually raw, full, and analyze  
 sep identity of the active assay  
 profile abundance  
 field taxonomy  
 common\_themes other meta.data  
 specific\_themes itol theme

---

itol.unit-class	<i>The itol.unit Class</i>
-----------------	----------------------------

---

### Description

The itol.unit object is an intermediate storage container used internally throughout the integration procedure to hold bits of data that are useful downstream.

### Slots

type a list of meta data table, usually raw, full, and analyze  
 sep identity of the active assay  
 profile abundance  
 field taxonomy  
 common\_themes other meta.data  
 specific\_themes itol theme  
 data data

---

learn_data	<i>Learn data from template file</i>
------------	--------------------------------------

---

### Description

Learn data from template file into data frame

### Usage

```
learn_data(df1 = NULL, file, tree = NULL, ...)
```

### Arguments

df1	initial data frame
file	template file
tree	tree file
...	Further arguments to be passed to subsequent functions.

### Value

a list with two data frame of node and tip annotation data

---

learn\_data\_from\_file *Learn object data from file*

---

**Description**

Learn itol.hub object data from template file.

**Usage**

```
learn_data_from_file(object, file)
```

**Arguments**

object	itol.hub object
file	template file

**Value**

a itol.hub object with new data from template file

---

learn\_data\_from\_files *Learn object data from files*

---

**Description**

Learn itol.hub object data from template file.

**Usage**

```
learn_data_from_files(object, files = NULL, dir = NULL, pattern = ".", ...)
```

**Arguments**

object	itol.hub object
files	template files path
dir	files path
pattern	file name pattern in regex
...	Further arguments to be passed to subsequent functions.

**Value**

a itol.hub object with new data from template files

---

learn\_data\_from\_unit    *Learn object data from unit*

---

**Description**

Learn itol.hub object data from unit object.

**Usage**

```
learn_data_from_unit(object, unit)
```

**Arguments**

object	itol.hub object
unit	itol.unit object

**Value**

a itol.hub object containing new data from itol.unit object

---

learn\_data\_from\_unit\_list  
*Learn object data from units*

---

**Description**

Learn itol.hub object data from list of unit object.

**Usage**

```
learn_data_from_unit_list(object, units)
```

**Arguments**

object	itol.hub object
units	itol.unit object list

**Value**

a itol.hub object with new data from a list of itol.unit objects

---

learn_df	<i>Learn from tree</i>
----------	------------------------

---

**Description**

Learn initial data frame from Newick format tree leaves.

**Usage**

```
learn_df(tree, node = FALSE, tip = TRUE)
```

**Arguments**

tree	Newick tree file or phylo object.
node	a logical to control output with node label or not. The default value is FALSE.
tip	a logical to control output tip label or not. The default value is TRUE.

**Value**

a list containing	
node	a data frame with id column. The id information is from the node label in Newick format tree file or phylo object. If the node parameter set as FALSE, the node information will be NULL.
tip	a data frame with id column. The id information is from the tip label in Newick format tree file or phylo object. If the tip parameter set as FALSE, the tip information will be NULL.

**Examples**

```
tree <- system.file("extdata",
                    "tree_of_itol_templates.tree",
                    package = "itol.toolkit")
sub_df <- learn_df(tree, node=TRUE, tip=TRUE)
```

---

learn_field	<i>Learn field</i>
-------------	--------------------

---

**Description**

learn field paramters as list

**Usage**

```
learn_field(lines, sep)
```

**Arguments**

lines            a vector of character strings from template file.  
 sep             a character specifying the separator.

**Value**

a list of field parameters containing

labels           a vector of characters specifying the field name. In DATASET\_HEATMAP, the labels are shown as heatmap column names.  
 colors          define colors for each individual field column (use hexadecimal, RGB or RGBA notation; if using RGB/RGBA, COMMA cannot be used as SEPARATOR)  
 shapes          Shape should be a number between 1 and 6, or any protein domain shape definition. 1-square, 2-circle, 3-star, 4-right pointing triangle, 5-left pointing triangle, 6-checkmark

**Examples**

```
tree <- system.file("extdata",
                    "tree_of_itol_templates.tree",
                    package = "itol.toolkit")
df_frequence <- data.table::fread(system.file("extdata",
                                             "templates_frequence.txt",
                                             package = "itol.toolkit"))

## create unit
unit <- create_unit(data = df_frequence,
                   key = "Quickstart",
                   type = "DATASET_HEATMAP",
                   tree = tree)

## write unit
file <- tempfile()
write_unit(unit, file)
## Learn legend parameters
lines <- line_clean(file=file)
sep = learn_separator(file = file)
learn_field(lines = lines, sep = sep)
```

---

 learn\_legend

*Learn legend*


---

**Description**

learn legend parameters as list

**Usage**

```
learn_legend(lines, sep)
```

**Arguments**

lines            a vector of character strings from template file.  
 sep             a character specifying the separator.

**Value**

a list of legned parameters containing

title            a character specifying the title of legend. There should not be the character same with separator within.  
 position\_x      a number specifying the x axis px value of the legend.  
 position\_y      a number specifying the y axis px value of the legend.  
 horizontal      To order legend entries horizontally instead of vertically, set this parameter to 1  
 shapes          Shape should be a number between 1 and 6, or any protein domain shape definition. 1-square, 2-circle, 3-star, 4-right pointing triangle, 5-left pointing triangle, 6-checkmark  
 colors          define colors for each legend element (use hexadecimal, RGB or RGBA notation; if using RGB/RGBA, COMMA cannot be used as SEPARATOR)  
 labels          The legend element label. There should not be the character same with separator within.  
 shape\_scales    For each shape, you can define a scaling factor between 0 and 1.

**Examples**

```
tree <- system.file("extdata",
  "tree_of_itol_templates.tree",
  package = "itol.toolkit")
df_frequence <- data.table::fread(system.file("extdata",
  "templates_frequence.txt",
  package = "itol.toolkit"))

## create unit
unit <- create_unit(data = df_frequence,
  key = "Quickstart",
  type = "DATASET_SIMPLEBAR",
  method = "mean",
  tree = tree)

## write unit
file <- tempfile()
write_unit(unit, file)
## Learn legend parameters
lines <- line_clean(file=file)
sep = learn_separator(file = file)
learn_legend(lines = lines, sep = sep)
```

---

learn_line	<i>Learn paramter</i>
------------	-----------------------

---

### Description

learn paramter name and values based on the key name in the front of line.

### Usage

```
learn_line(lines, param, sep)
```

### Arguments

lines	a vector of character strings from template file.
param	a charactor string of paramter key name. The key name should be uppercase letters or '_' without spacing.
sep	a charactor specifying the separator.

### Value

a charactor string containing parameter value.

### Examples

```
tree <- system.file("extdata",
                    "tree_of_itol_templates.tree",
                    package = "itol.toolkit")
data("template_groups")
df_group <- data.frame(id = unique(template_groups$group),
                      data = unique(template_groups$group))

## create unit
unit <- create_unit(data = df_group,
                   key = "Quickstart",
                   type = "DATASET_COLORSTRIP",
                   tree = tree)

## write unit
file <- tempfile()
write_unit(unit, file)
## Learn parameter
lines <- line_clean(file=file)
sep = learn_separator(file = file)
learn_line(lines = lines, param = "STRIP_WIDTH", sep = sep)
```



---

learn_profile	<i>Learn profile</i>
---------------	----------------------

---

**Description**

learn profile paramters as list

**Usage**

```
learn_profile(lines, sep)
```

**Arguments**

lines            a vector of character strings from template file.  
sep              a character specifying the separator.

**Value**

a list of profile parameters containing

name            a character specifying label, which is used in the legend table  
color           dataset color in the legend (use hexadecimal, RGB or RGBA notation; if using  
                 RGB/RGBA, COMMA cannot be used as SEPARATOR)

**Examples**

```
tree <- system.file("extdata",  
                    "tree_of_itol_templates.tree",  
                    package = "itol.toolkit")  
df_frequence <- data.table::fread(system.file("extdata",  
                                              "templates_frequence.txt",  
                                              package = "itol.toolkit"))  
  
## create unit  
unit <- create_unit(data = df_frequence,  
                    key = "Quickstart",  
                    type = "DATASET_HEATMAP",  
                    tree = tree)  
  
## write unit  
file <- tempfile()  
write_unit(unit, file)  
## Learn legend parameters  
lines <- line_clean(file=file)  
sep = learn_separator(file = file)  
learn_profile(lines = lines, sep = sep)
```

---

learn_separator	<i>Learn separator</i>
-----------------	------------------------

---

**Description**

Learn 3 types of separators: tab, space, and comma.

**Usage**

```
learn_separator(lines = NULL, file = NULL)
```

**Arguments**

lines	a vector of character strings from template file. If the file parameter is NULL, this parameter should be set.
file	a character specifying the template file path. If this parameter is setted, the lines parameter will be replaced.

**Value**

a character specifying the separator

**Examples**

```
tree <- system.file("extdata",
                   "tree_of_itol_templates.tree",
                   package = "itol.toolkit")
data("template_groups")
df_group <- data.frame(id = unique(template_groups$group),
                      data = unique(template_groups$group))

## create unit
unit <- create_unit(data = df_group,
                   key = "Quickstart",
                   type = "DATASET_COLORSTRIP",
                   tree = tree)

## write unit
file <- tempfile()
write_unit(unit, file)
## Learn template type
learn_separator(file = file)
```

---

learn_subdf	<i>Learn sub data frame</i>
-------------	-----------------------------

---

**Description**

Learn sub data frame from template file

**Usage**

```
learn_subdf(lines, type, sep, dataset_name = NULL, field_labels = NULL)
```

**Arguments**

lines	a vector of character strings from template file.
type	template type
sep	a character specifying the separator.
dataset_name	label in template file
field_labels	sample ids for binary, heatmap, and other multi-column value templates

**Value**

a data frame containing the data learned from template file

---

learn_theme_align	<i>Learn align</i>
-------------------	--------------------

---

**Description**

learn connection paramters as list

**Usage**

```
learn_theme_align(lines, sep)
```

**Arguments**

lines	a vector of character strings from template file.
sep	a character specifying the separator.

**Value**

a list of align parameters containing

---

learn\_theme\_alignment *Learn alignment*

---

**Description**

learn alignment paramters as list

**Usage**

```
learn_theme_alignment(lines, sep)
```

**Arguments**

lines	a vector of character strings from template file.
sep	a character specifying the separator.

**Value**

a list of alignment parameters containing

---

learn\_theme\_bar *Learn bar*

---

**Description**

learn bar paramters as list

**Usage**

```
learn_theme_bar(lines, sep)
```

**Arguments**

lines	file lines
sep	a character specifying the separator.

**Value**

a list of bar parameters containing

---

`learn_theme_basic_plot`*Learn basic plot*

---

**Description**

learn basic plot paramters as list

**Usage**

```
learn_theme_basic_plot(lines, sep)
```

**Arguments**

`lines` a vector of character strings from template file.  
`sep` a character specifying the separator.

**Value**

a list of basic plot parameters containing

---

`learn_theme_basic_theme`*Learn basic theme*

---

**Description**

learn basic theme paramters as list

**Usage**

```
learn_theme_basic_theme(lines, sep)
```

**Arguments**

`lines` a vector of character strings from template file.  
`sep` a character specifying the separator.

**Value**

a list of basic theme parameters containing

---

learn\_theme\_binary     *Learn binary*

---

**Description**

learn binary paramters as list

**Usage**

```
learn_theme_binary(lines, sep)
```

**Arguments**

lines                a vector of character strings from template file.  
sep                   a character specifying the separator.

**Value**

a list of binary chart parameters containing

---

learn\_theme\_border     *Learn border*

---

**Description**

learn border paramters as list

**Usage**

```
learn_theme_border(lines, sep)
```

**Arguments**

lines                a vector of character strings from template file.  
sep                   a character specifying the separator.

**Value**

a list of border parameters containing

---

`learn_theme_common_themes`*Learn common themes*

---

**Description**

learn common theme paramters as list

**Usage**

```
learn_theme_common_themes(lines, sep)
```

**Arguments**

`lines`            a vector of character strings from template file.  
`sep`              a character specifying the separator.

**Value**

a list of common theme parameters containing

---

`learn_theme_connection`*Learn connection*

---

**Description**

learn connection paramters as list

**Usage**

```
learn_theme_connection(lines, sep)
```

**Arguments**

`lines`            a vector of character strings from template file.  
`sep`              a character specifying the separator.

**Value**

a list of connection parameters containing

learn\_theme\_domain     *Learn domain*

---

**Description**

learn domain paramters as list

**Usage**

```
learn_theme_domain(lines, sep)
```

**Arguments**

lines            a vector of character strings from template file.  
sep              a character specifying the separator.

**Value**

a list of domain parameters containing

---

learn\_theme\_externalshape  
                          *Learn externalshape*

---

**Description**

learn connection paramters as list

**Usage**

```
learn_theme_externalshape(lines, sep)
```

**Arguments**

lines            a vector of character strings from template file.  
sep              a character specifying the separator.

**Value**

a list of external shape parameters containing



---

learn\_theme\_heatmap    *Learn heatmap*

---

**Description**

learn heatmap paramters as list

**Usage**

```
learn_theme_heatmap(lines, sep)
```

**Arguments**

lines            a vector of character strings from template file.  
sep              a character specifying the separator.

**Value**

a list of heatmap parameters containing

---

learn\_theme\_image    *Learn image*

---

**Description**

learn connection paramters as list

**Usage**

```
learn_theme_image(lines, sep)
```

**Arguments**

lines            a vector of character strings from template file.  
sep              a character specifying the separator.

**Value**

a list of image parameters containing



```

tab_id_group <- tab_tmp[,c(1,2)]
tab_tmp <- tab_tmp[,-c(1,2)]
tab_tmp_01 <- convert_01(object = tab_tmp)
tab_tmp_01 <- cbind(tab_id_group,tab_tmp_01)
order <- c("type","separator","profile","field","common themes",
  "specific themes","data")
tab_tmp_01_long <- tab_tmp_01 %>%
  tidyr::gather(key = "variable",
    value = "value",
    c(-parameter,-group))
template_start_group <- tab_tmp_01_long %>%
  group_by(group,variable) %>%
  summarise(sublen = sum(value)) %>%
  tidyr::spread(key=variable,
    value=sublen)
template_start_group$group <- factor(template_start_group$group,
  levels = order)
template_start_group <- template_start_group %>% arrange(group)
start_group <- data.frame(Var1 = template_start_group$group,
  Freq = apply(template_start_group[,-1], 1, max))
start_group$start <- 0
for (i in 2:nrow(start_group)) {
  start_group$start[i] <- sum(start_group$Freq[1:(i-1)])
}
template_start_group[template_start_group == 0] <- NA
template_end_group <- template_start_group[,2:(ncol(template_start_group)-1)] + start_group$start
template_end_group <- data.frame(group = order,template_end_group)
template_end_group_long <- template_end_group %>%
  tidyr::gather(key = "variable",
    value = "value",
    -group)
names(template_end_group_long)[3] <- "end"
template_end_group_long$start <- rep(start_group$start,
  length(unique(template_end_group_long$variable)))
template_end_group_long <- template_end_group_long %>% na.omit()
template_end_group_long$length <- sum(start_group$Freq)
template_end_group_long <- template_end_group_long[,c(2,5,4,3,1)]
template_end_group_long$group <- factor(template_end_group_long$group,levels = order)
unit <- create_unit(data = template_end_group_long,
  key = "Quickstart",
  type = "DATASET_DOMAINS",
  tree = tree)
file <- tempfile()
write_unit(unit,file)
lines <- line_clean(file=file)
sep = learn_separator(file = file)
learn_theme_label(lines,sep)

```

**Description**

learn linechart paramters as list

**Usage**

```
learn_theme_linechart(lines, sep)
```

**Arguments**

lines	a vector of character strings from template file.
sep	a character specifying the separator.

**Value**

a list of line chart parameters containing

---

learn\_theme\_piechart *Learn piechart*

---

**Description**

learn piechart paramters as list

**Usage**

```
learn_theme_piechart(lines, sep)
```

**Arguments**

lines	a vector of character strings from template file.
sep	a character specifying the separator.

**Value**

a list of pie chart parameters containing

---

`learn_theme_specific_themes`*Learn specific themes*

---

**Description**

learn specific theme paramters as list

**Usage**

```
learn_theme_specific_themes(lines, sep, type)
```

**Arguments**

lines            a vector of character strings from template file.

sep             a character specifying the separator.

type            template type

**Value**

a list of specific theme parameters containing

---

`learn_theme_strip_label`*Learn strip label*

---

**Description**

learn strip label paramters as list

**Usage**

```
learn_theme_strip_label(lines, sep)
```

**Arguments**

lines            a vector of character strings from template file.

sep             a charactor specifying the separator.

**Value**

a list of strip label parameters containing

display	0/1 specifying display or hide the individual label inside each colored strip (when defined in the data below)
width	a number specifying width of the colored strip
size	a number specifying strip label size factor (relative to the tree leaf labels)
color	define colors for each strip label element (use hexadecimal, RGB or RGBA notation; if using RGB/RGBA, COMMA cannot be used as SEPARATOR)
color_branches	1/0 specifying branches of the tree will or not be colored according to the colors of the strips above the leaves. When all children of a node have the same color, it will be colored the same, ie. the color will propagate inwards towards the root.
position	a character specifying position of the strip label within the box; 'top', 'center' or 'bottom'
shift	a number specifying strip label shift in pixels (positive or negative)
rotation	a number specifying rotation of the strip labels; used only in rectangular tree display mode
outline_width	a number specifying draw a black outline around the text (width in pixels)

**Examples**

```
tree <- system.file("extdata",
                   "tree_of_itol_templates.tree",
                   package = "itol.toolkit")
data("template_groups")
df_group <- data.frame(id = unique(template_groups$group),
                      data = unique(template_groups$group))

## create unit
unit <- create_unit(data = df_group,
                   key = "Quickstart",
                   type = "DATASET_COLORSTRIP",
                   tree = tree)

## write unit
file <- tempfile()
write_unit(unit, file)
## Learn parameter
lines <- line_clean(file=file)
sep = learn_separator(file = file)
learn_theme_strip_label(lines = lines, sep = sep)
```

---

learn\_type

*Learn template type*

---

**Description**

Extract first line of template to learn type information.

**Usage**

```
learn_type(file)
```

**Arguments**

**file** template file. All the template files should follow the format rules as same with iTOL official template files. The files should start with the following headers: "COLLAPSE", "PRUNE", "SPACING", "TREE\_COLORS", "DATASET\_STYLE", "LABELS", "DATASET\_TEXT", "DATASET\_COLORSTRIP", "DATASET\_BINARY", "DATASET\_GRADIENT", "DATASET\_HEATMAP", "DATASET\_SYMBOL", "DATASET\_EXTERNALSHAPE", "DATASET\_DOMAINS", "DATASET\_SIMPLEBAR", "DATASET\_MULTIBAR", "DATASET\_BOXPLOT", "DATASET\_LINECHART", "DATASET\_PIECHART", "DATASET\_ALIGNMENT", "DATASET\_CONNECTION", "DATASET\_IMAGE", "POPOPUP\_INFO".

**Value**

a character specifying header information

**Examples**

```
tree <- system.file("extdata",
                    "tree_of_itol_templates.tree",
                    package = "itol.toolkit")
data("template_groups")
df_group <- data.frame(id = unique(template_groups$group),
                      data = unique(template_groups$group))

## create unit
unit <- create_unit(data = df_group,
                    key = "Quickstart",
                    type = "DATASET_COLORSTRIP",
                    tree = tree)

## write unit
file <- tempfile()
write_unit(unit, file)
## Learn template type
learn_type(file)
```

---

line\_clean

*Filter out comments and empty lines*


---

**Description**

Remove the lines start with # or without any information.

**Usage**

```
line_clean(lines = NULL, file = NULL)
```

**Arguments**

- lines            a vector of character strings. The strings are containing the lines of template file. If the file parameter is NULL, this parameter should be set.
- file            a character specifying the template file path. If this parameter is setted, the lines parameter will be replaced.

**Value**

a vector of character strings

**Examples**

```
strs <- c("#comment", "DATA")
line_clean(lines=strs)
```

---

line_split	<i>Split lines into two parts</i>
------------	-----------------------------------

---

**Description**

Split lines based on the data block marker

**Usage**

```
line_split(lines, param = "data")
```

**Arguments**

- lines            a vector of character strings from template file.
- param            "theme" or "data" for the theme paramters or the data lines

**Value**

a vector of character strings containing data or theme information



---

`merge_unit`*Merge units*

---

**Description**

Merge two `itol.unit` with same type. The second unit data will be added into the first one.

**Usage**

```
merge_unit(obj1, obj2)
```

**Arguments**

<code>obj1</code>	a <code>itol.unit</code> object specifying the first unit
<code>obj2</code>	a <code>itol.unit</code> object specifying the second unit

**Value**

a `itol.unit` object with merged data

---

`show, itol.hub-method` *show method for S4 class itol.hub*

---

**Description**

show method for S4 class `itol.hub`

**Usage**

```
## S4 method for signature 'itol.hub'  
show(object)
```

**Arguments**

<code>object</code>	An object of class <code>itol.hub</code>
---------------------	--

**Value**

a stdout screen information about `itol.hub` object

---

template_groups	<i>template groups</i>
-----------------	------------------------

---

### Description

Templates were clustered into 5 groups by parameter similarity.

### Usage

template\_groups

### Format

template\_groups:

A data frame with template group clustering result:

**template** All the 23 template types of iTOL

**group 5** clustering groups: Tree structure: This group only controls the topology of tree branch merging, filtering, and spacing. There are no style and rich annotation data, even though most of the annotation data only include single-column id information and do not contain any dataset base information, sample information, or common and specific style information. It is a particularly simple type of template. Theme style: This does not change any topology or add any text information but only changes the color scheme, line type and width, and font style and size of existing information. This is an extremely comprehensive and diverse type of annotation information. Text: This group contains any templates with added text information. With super flexible and convenient annotation methods, users can modify even a single character's style in HTML. Users can also modify the text annotation style of nodes and branches in batch based on matching conditions in itol.hub objects, which require regular expression replacement and precise data filtering. This high-frequency data processing is difficult to achieve and retain the workflow in the EXCEL-based editor. Basic plot: This group contains basic visualization methods. From a functional point of view, this is the most feature-rich class of templates. The similarity of the parameters within this part is very high. The structured and uniform organization of these templates can greatly reduce code redundancy and the user workload of data organizing. Moreover, boxplot, which is not a regular enough data annotation template, can be automatically manipulated in R. The lack of template data structure makes using frequency unbalanced among research. Hence, the frequency of using these low-frequency templates can be increased. Advanced plot: Compared with the basic visualization methods, these visualization methods contain more comprehensive data types and often require third-party tools for input data processing. But they are the most extensible type of visualization methods for iTOL.

...

---

template_parameters_count	<i>template parameters count</i>
---------------------------	----------------------------------

---

**Description**

Template types and parameters count matrix. The row names are template types. The column names are parameters short ids. The parameters are including the themes parameters and data column names. All the details are introduced in the full-page Excel file on GitHub.

**Usage**

```
template_parameters_count
```

**Format**

template\_parameters\_count:

A data frame with template types and parameters 0/1 count matrix:

**V1** head. file type head notice

**V2** separator. select the separator which is used to delimit the data below (TAB,SPACE or COMMA).This separator must be used throughout this file.

**V3** dataset name. label is used in the legend table ...

---

unite_rows	<i>Paste rows</i>
------------	-------------------

---

**Description**

Paste rows group by key column

**Usage**

```
unite_rows(df)
```

**Arguments**

df                   input data frame

**Value**

a data frame with pasted row by same id

---

use.theme	<i>Extract theme from inbuilt_themes</i>
-----------	--

---

### Description

Extract theme from 23 template types in inbuilt\_themes data in package.

### Usage

```
use.theme(type, style = "default")
```

### Arguments

type	a character specifying the template type used for extracting. Following choices are possible: "COLLAPSE", "PRUNE", "SPACING", "TREE_COLORS", "DATASET_STYLE", "LABEL", "DATASET_BINARY", "DATASET_GRADIENT", "DATASET_HEATMAP", "DATASET_SYMBOL", "DATASET_EXTERNALSHAPE", "DATASET_DOMAINS", "DATASET_SIMPLEBAR", "DATASET_MULTIBAR", "DATASET_BOXPLOT", "DATASET_LINECHART", "DATASET_PIECHART", "DATASET_ALIGNMENT", "DATASET_CONNECTION", "DATASET_IMAGE", "POPUP_INFO.
style	a character specifying the specific version of template type used for extracting. The default value is "default" style for all types.

### Value

a itol.theme object containing

type	This group holds information about the template type of the data only. This is a very critical piece of information. In many functions of the itol.toolkit package, the template type information is used to determine the different data processing and input/output methods.
sep	This group holds data separator information only. This is one of the most important parameters for data reading and output. It is a separate category because it is frequently used and is an input parameter for other subsequent parameters to be read.
profile	This group contains basic information about the dataset, such as the dataset name and a color label to distinguish the dataset. The dataset name is extremely important. This parameter is used almost throughout the data processing of the itol.toolkit package. With the content of this parameter as the key value, the data and theme information of the dataset are associated. In turn, high throughput learning and writing of large-scale data can be achieved. This parameter is not included in some template types with a particularly simple structure, so we choose a file name or a user-defined method as the key value.
field	This group contains information about each sample within the dataset, and this type of parameter exists only for multi-sample data. This information even includes the clustering tree between samples. This information is usually stored as part of the column names in the metadata part or abundance information of the itol.hub object.

- `common_themes` These themes are used at high frequency in different templates. These parameters are small in number but constitute some common features of iTOL visual style settings, such as legend, margin, etc.
- `specific_themes` These themes are used only in specific templates. The number of these parameters is very large. However, most of them are used in only one template to control the style details of the visualization. By unifying these parameters and calling them according to the template type, users can perform secondary development and data processing with a high degree of parameter aggregation without worrying too much about the differences between different template types.

## Examples

```
theme <- use.theme("COLLAPSE")
```

---

<code>write_hub</code>	<i>Write all data object into files</i>
------------------------	---

---

## Description

Write `itol.hub` object into template files.

## Usage

```
write_hub(object, dir = getwd())
```

## Arguments

- `object` `itol.hub` object holds the complete data and theme information. This is an all-in-one object that collects all the information. Based on this object, it is possible to export template files directly. It can also be converted to an operation unit object for the detailed processing of individual datasets. The object can also be saved locally for reproducible visualization to share. This object contains species or sample clustering trees, sequence alignment, species abundance or gene expression table, multi-level taxonomic information, metadata, and a list of custom themes. Each element name in the theme list is prefixed with the column name of the metadata and is used to establish the association between the theme and the data. For some special dataset types, the storage location is not in the metadata, but it also conforms to the association with themes. The program automatically decides where to read the data according to the different output template types. The user only needs to explicitly define the theme name to be output consistent with the data name prefix.
- `dir` output dir path. Define the output files location using absolute or relative path. The template files will output by the key information from theme name in the hub object.

**Value**

No return value, only output template files

**Examples**

```
tree <- system.file("extdata",
                    "tree_of_itol_templates.tree",
                    package = "itol.toolkit")
hub <- create_hub(tree = tree)
data("template_groups")
df_group <- data.frame(id = unique(template_groups$group),
                      data = unique(template_groups$group))

## create unit
unit_1 <- create_unit(data = df_group,
                     key = "Quickstart_1",
                     type = "TREE_COLORS",
                     subtype = "clade",
                     line_type = c(rep("normal", 4), "dashed"),
                     size_factor = 5,
                     tree = tree)
unit_2 <- create_unit(data = df_group,
                     key = "Quickstart_2",
                     type = "DATASET_COLORSTRIP",
                     tree = tree)

## write hub
hub <- hub + unit_1 + unit_2
write_hub(hub, tempdir())
```

---

write\_raw

*Write raw data into files*

---

**Description**

Write raw data in itol.hub object into files

**Usage**

```
write_raw(object, dir, title)
```

**Arguments**

**object** itol.hub object holds the complete data and theme information. This is an all-in-one object that collects all the information. Based on this object, it is possible to export template files directly. It can also be converted to an operation unit object for the detailed processing of individual datasets. The object can also be saved locally for reproducible visualization to share. This object contains species or sample clustering trees, sequence alignment, species abundance or gene expression table, multi-level taxonomic information, metadata, and a list of custom themes. Each element name in the theme list is prefixed with the

column name of the metadata and is used to establish the association between the theme and the data. For some special dataset types, the storage location is not in the metadata, but it also conforms to the association with themes. The program automatically decides where to read the data according to the different output template types. The user only needs to explicitly define the theme name to be output consistent with the data name prefix.

**dir** output dir path. Define the output files location using absolute or relative path. The raw data will write into files. The following raw data will be outputted: main tree, sample tree, alignment sequences, abundance count table, taxonomy table, metadata on nodes and tips.

**title** files name title string. This character specified the prefix of raw data files.

### Value

No return value, only output raw data files

### Examples

```
tree <- system.file("extdata",
                   "tree_of_itol_templates.tree",
                   package = "itol.toolkit")
hub <- create_hub(tree = tree)
df_values <- data.table::fread(system.file("extdata",
                                          "templates_frequence.txt",
                                          package = "itol.toolkit"))

unit <- create_unit(data = df_values,
                   key = "Quickstart",
                   type = "DATASET_HEATMAP",
                   tree = tree)

hub <- hub + unit
write_raw(hub, tempdir(), "Quickstart")
```

---

write\_unit

*Write unit object into file*

---

### Description

Write itol.unit object into template file. This function will using the type information in unit object to decide different output methods for the template formats.

### Usage

```
write_unit(unit, file = getwd())
```

**Arguments**

unit	unit object. The unit object holds the data and theme of a single dataset. This is the smallest data operation unit. At this level, individual data can be fine-tuned. It is also possible to extract the style of a unit for use in other units. It is also possible to use many units to learn a complete itol.hub object. Almost all specific data operations behind the itol.toolkit package are performed at the unit level. Because itol.hub objects have comprehensive information, but to ensure that the correspondence with phylogenetic branches or nodes remains consistent when different data types are saved, many complex data aggregations are saved, which does not facilitate data processing. Therefore, in the actual data processing process, unit objects are generated from the itol.hub object and then processed.
file	output file path. Define the output file location and file name using absolute or relative path.

**Value**

No return value, only output a template file

**Examples**

```
tree <- system.file("extdata",
                    "tree_of_itol_templates.tree",
                    package = "itol.toolkit")
data("template_groups")
df_group <- data.frame(id = unique(template_groups$group),
                      data = unique(template_groups$group))
## create unit
unit <- create_unit(data = df_group,
                   key = "Quickstart",
                   type = "DATASET_COLORSTRIP",
                   tree = tree)
## write unit
write_unit(unit, tempfile())
```



# Index

- \* **datasets**
  - inbuilt\_themes, 16
  - template\_groups, 42
  - template\_parameters\_count, 43
- \* **objects**
  - itol.hub-class, 17
  - itol.theme-class, 17
  - itol.unit-class, 18
- \* **object**
  - create\_hub, 7
  - create\_theme, 8
  - file\_to\_unit, 14
  - hub\_to\_unit, 15
- +, itol.hub, itol.unit-method, 3
- +, itol.unit, itol.unit-method
  - (+, itol.hub, itol.unit-method), 3
- complex\_html\_text, 4
- convert\_01, 4
- convert\_01\_to\_connect, 5
- convert\_range\_to\_node, 5
- correct\_get\_color, 6
- count\_to\_tree, 6
- create\_hub, 7
- create\_theme, 8
- create\_unit, 8
- df\_merge, 11
- fa\_read, 12
- fa\_write, 12
- file\_get\_dir, 13
- file\_get\_name, 13
- file\_to\_unit, 14
- get\_color, 14
- head\_line, 15
- hub\_to\_unit, 15
- inbuilt\_themes, 16
- itol.hub (itol.hub-class), 17
- itol.hub-class, 17
- itol.theme (itol.theme-class), 17
- itol.theme-class, 17
- itol.unit (itol.unit-class), 18
- itol.unit-class, 18
- learn\_data, 18
- learn\_data\_from\_file, 19
- learn\_data\_from\_files, 19
- learn\_data\_from\_unit, 20
- learn\_data\_from\_unit\_list, 20
- learn\_df, 21
- learn\_field, 21
- learn\_legend, 22
- learn\_line, 24
- learn\_profile, 25
- learn\_separator, 26
- learn\_subdf, 27
- learn\_theme\_align, 27
- learn\_theme\_alignment, 28
- learn\_theme\_bar, 28
- learn\_theme\_basic\_plot, 29
- learn\_theme\_basic\_theme, 29
- learn\_theme\_binary, 30
- learn\_theme\_border, 30
- learn\_theme\_common\_themes, 31
- learn\_theme\_connection, 31
- learn\_theme\_domain, 32
- learn\_theme\_externalshape, 32
- learn\_theme\_heatmap, 33
- learn\_theme\_image, 33
- learn\_theme\_label, 34
- learn\_theme\_linechart, 35
- learn\_theme\_piechart, 36
- learn\_theme\_specific\_themes, 37
- learn\_theme\_strip\_label, 37
- learn\_type, 38
- line\_clean, 39

`line_split`, [40](#)

`merge_unit`, [41](#)

`show`, `itol.hub-method`, [41](#)

`template_groups`, [42](#)

`template_parameters_count`, [43](#)

`unite_rows`, [43](#)

`use.theme`, [44](#)

`write_hub`, [45](#)

`write_raw`, [46](#)

`write_unit`, [47](#)