Package 'tmplate'

October 14, 2022

Type Package

Title Code Generation Based on Templates

Version 0.0.3

Date 2021-07-19

Author Mario A. Martinez Araya [aut, cre, cph]
(<https://orcid.org/0000-0002-4821-9314>)

Maintainer Mario A. Martinez Araya <r@marioma.me>

Description Define general templates with tags that can be replaced by content depending on arguments and objects to modify the final output of the document.

License GPL (>= 2)

Depends R (>= 2.10), tRnslate

Suggests knitr, rmarkdown, markdown

URL <https://marioma.me?i=soft>

BuildVignettes yes

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2021-07-20 07:50:02 UTC

R topics documented:

tmplate-package *'Code Generation Based on Templates'*

Description

Define general templates with tags that can be replaced by content depending on arguments and objects to modify the final output of the document.

Details

Function `translate` receives a template (as a character vector with the lines of a template file which contains tags either for content or R code) a list of argument variables that will be used to translate those tags to content. The evaluation is performed in an environment defined by the user and replaces its output in the template returning a character vector with the lines of the resulting template.

Examples

```
library(tmplate)
#
### use the following template file
T <- readLines(system.file("examples/template.txt", package = "tmplate"))
#
### translate template content using the following values as input arguments
TT <- translate(
SHELL_CALL='#!/bin/bash',
SLURM_SBATCH=ifelse(.Platform$OS.type=="unix",
ifelse(system("clu=$(sinfo --version 2>&1) || clu=`echo -1`; echo $clu", intern = TRUE)=="-1",
'<:MISS:>','#SBATCH '),'<:MISS:>'),
SLURM_PARTITION='<:SLURM_SBATCH:>--partition=defq',
SLURM_ASK_NODES=2,
SLURM_NODES='<:SLURM_SBATCH:>--nodes=<:SLURM_ASK_NODES:>',
SLURM_ASK_TASKS=4,
SLURM_TASKS='<:SLURM_SBATCH:>--ntasks-per-node=<:SLURM_ASK_TASKS:>',
SLURM_MEMORY='<:SLURM_SBATCH:>--memory=2gb',
SLURM_TIME='<:SLURM_SBATCH:>--time=1:00:00',
SLURM_ARRAY="<:MISS:>",
MODULES_LOAD='module load module/for/openmpi module/for/R',
WORKDIR=ifelse('<:SLURM_SBATCH:>'!='#SBATCH','# no slurm machine','cd ${SLURM_SUBMIT_DIR}'),
TASK="<:MISS:>",
PASS_TASK="<:MISS:>",
PASS_TASK_VAR="<:MISS:>",
MPI_N="<:MISS:>",
MPI_ASK_N='<r@ <:SLURM_ASK_NODES:> * <:SLURM_ASK_TASKS:> @>',
R_HOME=R.home("bin"),
R_OPTIONS='--no-save --no-restore',
R_FILE_INPUT='script.R',
R_ARGS='',
R_FILE_OUTPUT='output.Rout',
MPIRUN='mpirun --mca mpi_warn_on_fork 0 -n <:MPI_ASK_N:> /
```

```
<:R_HOME:>/Rscript <:R_OPTIONS:> "<:R_FILE_INPUT:>" /
<r@ ifelse(!any(grepl("^<:MISS:>$","<:SLURM_ARRAY:>")),"<:PASS_TASK_VAR:>","") @> /
<:R_ARGS:> > <:R_FILE_OUTPUT:>',
MESSAGE_CLOSE='echo "Job submitted on $(date)."',
drop = TRUE,
default = "MISS",
template = T
)
#
### run TT to display resulting output (or print using 'cat' with newline as separator)
TT
#
### Or See instructions in the following file:
### (example present in the vignette)
source(system.file("examples/example.R", package = "tmplate"))
```

`translate` *Translate tags and R code in template*

Description

Evaluate tags and inline or chunks of R code present in general template files to produce variable content depending on some input arguments.

Usage

```
translate(vars, ..., template, drop = FALSE, default = "NULL",
        warn = TRUE, start = "<:", end = ":>", redo = 2,
        envir = new.env(parent = parent.frame()), retrans = 2,
        debug = FALSE)
```

Arguments

`vars`	list with named elements containing arbitrary variables to search in templates. To specify variables should be used one of the arguments `vars` or `...` but not both. If both are defined only `vars` is used.
`...`	arbitrary variables to search in templates.
`template`	path to template to evaluate.
`drop`	drop lines with non evaluated variables? Default to FALSE.
`default`	default character to translate for not given or missing variables. Default to "NULL" (will appear preceded by "<:" and ":>" or `startNULLend` in translated file).
`warn`	print warning if some variables are still not translated in the finel file. Default to TRUE. If FALSE process will stop returning an error.
`start`	line number where the R code chunk starts, or string to match start of variable. Default to "<:".

end	line number where the R code chunk ends, or string to match end of variable. Default to ":>".
redo	how many times translate? Default to 2 (so variables can appear in definition of other variables and still be translated).
retrans	retranslate a template a number of times.
envir	environment where to evaluate R code.
debug	logical. Default to FALSE (no debug).

Details

The function `translate` evaluate 'tags' and inline or chunks of R code in a template modifying its content. The tags are calls that point to argument variables which are provided by the user. Thus, the main input arguments for `translate` are the template (provided as a character vector where each element correspond to a line in the template file) and the variables which will be used to evaluate the tags in the template to update its content. The evaluation is performed in an environment which can also be defined by the user. The output of `translate` is a character vector where each element correspond to the line of the modified template. The tags names are denoted in betweed `<: :>`, for example `<:tag_name:>`, and they must be placed in the position where we want their evaluation to occur and output to appear. We should provide `tag_name = value` in the arguments of function `translate`. The inline or chunks of R code must follow the rules set in translate_r_code. See tRnslate package vignette or run `vignette(tRnslate)`. For more details see tmplate package vignette or run `vignette("tmplate")`. For an example see tmplate-package.

Value

Once tags and the chunks or inline R code are evaluated by `translate` using the input arguments given, then it returns a character vector where each element corresponds to the original line in the template file where the tags, chunks and inline code has been replaced by its output. This content can be seen in console or written to disc, for example, by using `cat` (it requires to use `sep = "\n"`).

Author(s)

Mario A. Martinez Araya, <r@marioma.me>

Examples

```
## To see an example in R console run:
##
## ?tmplate::tmplate
##
## Or See instructions in the following file:
## (example present in the vignette)
source(system.file("examples/example.R", package = "tmplate"))
```

Index