# Package ‘templr’

August 31, 2022

**Type** Package

**Title** MASCOTNUM Algorithms Template Tools

**Version** 0.1-0

**Date** 2022-08-05

**Author** Yann Richet [aut, cre] (<https://orcid.org/0000-0002-5677-8458>)

**Maintainer** Yann Richet <yann.richet@irsn.fr>

**Description** Helper functions for MASCOTNUM algorithm template, for design of numerical experiments practice:
algorithm template parser to support MASCOTNUM specification <https://www.gdr-mascotnum.fr/template.html>,
'ask & tell' decoupling injection (inspired by <https://search.r-project.org/CRAN/refmans/sensitivity/html/decoupling.html>)
to use ``crimped'' algorithms (like uniroot(), optim(), ...) from outside R,
basic template examples: Brent algorithm for 1 dim root finding and L-BFGS-B from base optim().

**License** Apache License (>= 2)

**Encoding** UTF-8

**Depends** R (>= 4.0)

**Imports** utils, stats, remotes

**Suggests** testthat, future

**URL** https://github.com/MASCOTNUM/templr

**RoxygenNote** 7.2.1

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-08-31 19:00:01 UTC

## R topics documented:

## ask_dX — *ask&tell component function to 'ask' where objective function gradient evaluation is required.*

### Description

ask&tell component function to 'ask' where objective function gradient evaluation is required.

### Usage

```
ask_dX(
  id = 0,
  dX.tmp = "dX.todo",
  tmp_path = file.path(tempdir(), "..", "asktell.tmp"),
  sleep_step = 0.1,
  sleep_init = 0,
  timeout = 360000,
  trace = function(...) cat(paste0(..., "\n")),
  clean = TRUE
)
```

### Arguments

| | |
|---|---|
| `id` | unique identifier for this asktell loop (default: "0") |
| `dX.tmp` | temporary "X" values file (default: "dX.todo") |
| `tmp_path` | temporary directory to store X.tmp & Y.tmp (default: 'tempdir()/../asktell.tmp') |
| `sleep_step` | delay between checking X.tmp and Y.tmp (default: 0.1 sec.) |
| `sleep_init` | initial delay before checking X.tmp and Y.tmp (default: 0 sec.) |
| `timeout` | maximum delay before breaking loop if X.tmp or Y.tmp doesn't appear (default: 36000 sec. = 10 min.) . |
| `trace` | function to display asktell loop status (default : 'cat') |
| `clean` | should we cleanup temporary files after reading ? (default: TRUE) |

## Details

'ask&tell' injection loop to call an external objective function within an inline algorithm (like optim(...)) Main idea: pass 'ask_Y' as objectve function argument of algorithm, which will wait until you call 'tell_Y' in another R process. In this secondary process, you can read what X is called using 'ask_X', and when you know what values returns from the external objective, just call 'tell_Y' to give it.

## Value

input values of objective function to compute externally

## Author(s)

Y. Richet, discussions with D. Sinoquet. Async IO principle was defined by G. Pujol.

## Examples

```
## Not run:  ### Assumes you can use two independant R sessions
## In main R session
  ask_dY(x=123)
## In another R session
  ask_dX() # returns 123
  tell_dY(y=456)
## Then ask_dY in main R session returns with value '456'

## End(Not run)
```

---

`ask_dY` *ask&tell component function to 'ask' objective function gradient evaluation using finite difference.*

---

## Description

ask&tell component function to 'ask' objective function gradient evaluation using finite difference.

## Usage

```
ask_dY(
  x,
  dX = 0.001,
  id = 0,
  dX.tmp = "dX.todo",
  dY.tmp = "dY.done",
  tmp_path = file.path(tempdir(), "..", "asktell.tmp"),
  sleep_step = 0.1,
  sleep_init = 0,
  timeout = 360000,
  trace = function(...) cat(paste0(..., "\n")),
  clean = TRUE,
  force_cleanup = FALSE
)
```

## Arguments

| | |
|---|---|
| `x` | input values of objective function gradient to compute |
| `dX` | finite difference applied to input values to compute gradient |
| `id` | unique identifier for this asktell loop (default: "0") |
| `dX.tmp` | temporary "X" values file (default: "dX.todo") |
| `dY.tmp` | temporary "Y" values file (default: "dY.done") |
| `tmp_path` | temporary directory to store X.tmp & Y.tmp (default: 'tempdir()/../asktell.tmp') |
| `sleep_step` | delay between checking X.tmp and Y.tmp (default: 0.1 sec.) |
| `sleep_init` | initial delay before checking X.tmp and Y.tmp (default: 0 sec.) |
| `timeout` | maximum delay before breaking loop if X.tmp or Y.tmp doesn't appear (default: 36000 sec. = 10 min.) . |
| `trace` | function to display asktell loop status (default : 'cat') |
| `clean` | should we cleanup temporary files after reading ? (default: TRUE) |
| `force_cleanup` | should we cleanup temporary files before writing (possible conflicting asktell calls) ? (default: FALSE) |

## Details

'ask&tell' injection loop to call an external objective function within an inline algorithm (like optim(...)) Main idea: pass 'ask_Y' as objectve function argument of algorithm, which will wait until you call 'tell_Y' in another R process. In this secondary process, you can read what X is called using 'ask_X', and when you know what values returns from the external objective, just call 'tell_Y' to give it.

## Value

output value of objective function gradient, as given by tell_dY() call in parallel session

## Author(s)

Y. Richet, discussions with D. Sinoquet. Async IO principle was defined by G. Pujol.

## Examples

```
## Not run:  ### Assumes you can use two independant R sessions
## In main R session
  ask_dY(x=123)
## In another R session
  ask_dX() # returns 123
  tell_dY(y=456)
## Then ask_dY in main R session returns with value '456'

## End(Not run)
```

## ask_X *ask&tell component function to 'ask' where objective function evaluation is required.*

### Description

ask&tell component function to 'ask' where objective function evaluation is required.

### Usage

```
ask_X(
  id = 0,
  X.tmp = "X.todo",
  tmp_path = file.path(tempdir(), "..", "asktell.tmp"),
  sleep_step = 0.1,
  sleep_init = 0.1,
  timeout = 360000,
  trace = function(...) cat(paste0(..., "\n")),
  clean = TRUE
)
```

### Arguments

| | |
|---|---|
| `id` | unique identifier for this asktell loop (default: "0") |
| `X.tmp` | temporary "X" values file (default: "X.todo") |
| `tmp_path` | temporary directory to store X.tmp & Y.tmp (default: 'tempdir()/../asktell.tmp') |
| `sleep_step` | delay between checking X.tmp and Y.tmp (default: 0.1 sec.) |
| `sleep_init` | initial delay before checking X.tmp and Y.tmp (default: 0 sec.) |
| `timeout` | maximum delay before breaking loop if X.tmp or Y.tmp doesn't appear (default: 36000 sec. = 10 min.) . |
| `trace` | function to display asktell loop status (default : 'cat') |
| `clean` | should we cleanup temporary files after reading ? (default: TRUE) |

### Details

'ask&tell' injection loop to call an external objective function within an inline algorithm (like optim(...)) Main idea: pass 'ask_Y' as objectve function argument of algorithm, which will wait until you call 'tell_Y' in another R process. In this secondary process, you can read what X is called using 'ask_X', and when you know what values returns from the external objective, just call 'tell_Y' to give it.

### Value

input value of objective function to compute externally

### Author(s)

Y. Richet, discussions with D. Sinoquet. Async IO principle was defined by G. Pujol.

### Examples

```
## Not run:  ### Assumes you can use two independant R sessions
## In main R session
  ask_Y(x=123)
## In another R session
  ask_X() # returns 123
  tell_Y(y=456)
## Then ask_dY in main R session returns with value '456'

## End(Not run)
```

---

## ask_Y *ask&tell component function to 'ask' objective function evaluation.*

---

### Description

ask&tell component function to 'ask' objective function evaluation.

### Usage

```
ask_Y(
  x,
  id = 0,
  X.tmp = "X.todo",
  Y.tmp = "Y.done",
  tmp_path = file.path(tempdir(), "..", "asktell.tmp"),
  sleep_step = 0.1,
  sleep_init = 0,
  timeout = 360000,
  trace = function(...) cat(paste0(..., "\n")),
  clean = TRUE,
  force_cleanup = FALSE
)
```

### Arguments

| | |
|---|---|
| `x` | input values of objective function to compute |
| `id` | unique identifier for this asktell loop (default: "0") |
| `X.tmp` | temporary "X" values file (default: "X.todo") |
| `Y.tmp` | temporary "Y" values file (default: "Y.done") |
| `tmp_path` | temporary directory to store X.tmp & Y.tmp (default: 'tempdir()/../asktell.tmp') |
| `sleep_step` | delay between checking X.tmp and Y.tmp (default: 0.1 sec.) |

| | |
|---|---|
| `sleep_init` | initial delay before checking X.tmp and Y.tmp (default: 0 sec.) |
| `timeout` | maximum delay before breaking loop if X.tmp or Y.tmp doesn't appear (default: 36000 sec. = 10 min.) . |
| `trace` | function to display asktell loop status (default : 'cat') |
| `clean` | should we cleanup temporary files after reading ? (default: TRUE) |
| `force_cleanup` | should we cleanup temporary files before writing (possible conflicting asktell calls) ? (default: FALSE) |

## Details

'ask&tell' injection loop to call an external objective function within an inline algorithm (like optim(...)) Main idea: pass 'ask_Y' as objectve function argument of algorithm, which will wait until you call 'tell_Y' in another R process. In this secondary process, you can read what X is called using 'ask_X', and when you know what values returns from the external objective, just call 'tell_Y' to give it.

## Value

output value of objective function, as given by tell_Y() call in parallel session

## Author(s)

Y. Richet, discussions with D. Sinoquet. Async IO principle was defined by G. Pujol.

## Examples

```
## Not run:  ### Assumes you can use two independant R sessions
## In main R session
  ask_Y(x=123)
## In another R session
  ask_X() # returns 123
  tell_Y(y=456)
## Then ask_Y in main R session returns with value '456'

## End(Not run)
```

---

## `from01` *Helper function to scale from [0,1] to [min,max]*

---

## Description

Helper function to scale from [0,1] to [min,max]

## Usage

```
from01(X, inp)
```

## Arguments

| | |
|---|---|
| `X` | values to scale |
| `inp` | list containing 'min' and 'max' values |

## Value

X scaled in [inp$min, inp$max]

## Examples

```
from01(data.frame(x=matrix(runif(10))),list(x=list(min=10,max=20)))
```

---

## `import` *Dependencies loader, supports many protocols like github:, gitlab:, ... using remotes::instal_... Will create a local '.lib' directory to store packages installed*

---

## Description

Dependencies loader, supports many protocols like github:, gitlab:, ... using remotes::instal_... Will create a local '.lib' directory to store packages installed

## Usage

```
import(..., lib.loc = NULL, trace = function(...) cat(paste0(..., "\n")))
```

## Arguments

| | |
|---|---|
| `...` | dependencies/libraries/packages to load |
| `lib.loc` | use to setup a dedicated libPath directory to install packages |
| `trace` | display info |

## Value

(list of) load status of packages (TRUE/FALSE)

## Examples

```
if(interactive()){
  import('VGAM')
}
```

## max_input *Helper function to get $max from 'input' list*

### Description

Helper function to get $max from 'input' list

### Usage

```
max_input(inp)
```

### Arguments

| | |
|---|---|
| inp | lst of objects containing 'max' field (as list) |

### Value

array of inp$...$max values

### Examples

```
max_input(list(x1=list(min=0,max=1),x2=list(min=2,max=3)))
```

## min_input *Helper function to get $min from 'input' list*

### Description

Helper function to get $min from 'input' list

### Usage

```
min_input(inp)
```

### Arguments

| | |
|---|---|
| inp | lst of objects containing 'min' field (as list) |

### Value

array of inp$...$min values

### Examples

```
min_input(list(x1=list(min=0,max=1),x2=list(min=2,max=3)))
```

## `parse.algorithm` *Parse algorithm file and returns its (header) indos and methods*

### Description

Parse algorithm file and returns its (header) indos and methods

### Usage

```
parse.algorithm(file)
```

### Arguments

| | |
|---|---|
| `file` | Template algorithm file to parse |

### Value

list of header infos and environment containing methods <constructor>,getInitialDesign,getNextDesign,displayResults

### Examples

```
parse.algorithm(system.file("Brent.R", package="templr"))
```

## `read.algorithm` *Read algorithm file and returns one header info*

### Description

Read algorithm file and returns one header info

### Usage

```
read.algorithm(file, info = "help")
```

### Arguments

| | |
|---|---|
| `file` | Template algorithm file to read |
| `info` | header info to return |

### Value

list of header infos

### Examples

```
read.algorithm(system.file("Brent.R", package="templr"),"help")
```

## run.algorithm *Apply a template algorithm file to an objective function*

### Description

Apply a template algorithm file to an objective function

### Usage

```
run.algorithm(
  algorithm_file,
  objective_function,
  input,
  options = NULL,
  work_dir = ".",
  trace = function(...) cat(paste0(..., "\n")),
  silent = FALSE
)
```

### Arguments

| | |
|---|---|
| `algorithm_file` | tempalted algorithm file |
| `objective_function` | function to apply algorithm on |
| `input` | list of input arguments of function (eg. list(x1=list(min=0,max=1),x2=list(min=10,max=20))) |
| `options` | algorithm options to overload default ones |
| `work_dir` | working directory to run algorithm. will store output files, images, .. |
| `trace` | display running info |
| `silent` | quietness |

### Value

algorithm result (and algorithm object & files as attributes)

### Examples

```
run.algorithm(
  system.file("Brent.R", package="templr"),
  function(x) sin(x)-0.75,
  list(x=list(min=0,max=pi/2)),
  work_dir=tempdir()
  )
```

## tell_dY *ask&tell component function to 'tell' objective function value to waiting 'ask_Y' call in another R session.*

### Description

ask&tell component function to 'tell' objective function value to waiting 'ask_Y' call in another R session.

### Usage

```
tell_dY(
  dy,
  id = 0,
  dY.tmp = "dY.done",
  tmp_path = file.path(tempdir(), "..", "asktell.tmp"),
  trace = function(...) cat(paste0(..., "\n")),
  force_cleanup = FALSE
)
```

### Arguments

| | |
|---|---|
| dy | output value of objective function gradient to return |
| id | unique identifier for this asktell loop (default: "0") |
| dY.tmp | temporary "Y" values file (default: "dY.done") |
| tmp_path | temporary directory to store X.tmp & Y.tmp (default: 'tempdir()/../asktell.tmp') |
| trace | function to display asktell loop status (default : 'cat') |
| force_cleanup | should we cleanup temporary files before writing (possible conflicting asktell calls) ? (default: FALSE) |

### Details

'ask&tell' injection loop to call an external objective function within an inline algorithm (like optim(...)) Main idea: pass 'ask_Y' as objectve function argument of algorithm, which will wait until you call 'tell_Y' in another R process. In this secondary process, you can read what X is called using 'ask_X', and when you know what values returns from the external objective, just call 'tell_Y' to give it.

### Value

input value of objective function to compute externally

### Author(s)

Y. Richet, discussions with D. Sinoquet. Async IO principle was defined by G. Pujol.

## Examples

```
## Not run:  ### Assumes you can use two independant R sessions
## In main R session
  ask_dY(x=123)
## In another R session
  ask_dX() # returns c(123, 123.123)
  tell_dY(dy=c(456,456.123))
## Then ask_dY in main R session returns with value '1'

## End(Not run)
```

---

| | |
|---|---|
| `tell_Y` | *ask&tell component function to 'tell' objective function value to waiting 'ask_Y' call in another R session.* |

---

## Description

ask&tell component function to 'tell' objective function value to waiting 'ask_Y' call in another R session.

## Usage

```
tell_Y(
  y,
  id = 0,
  Y.tmp = "Y.done",
  tmp_path = file.path(tempdir(), "..", "asktell.tmp"),
  trace = function(...) cat(paste0(..., "\n")),
  force_cleanup = FALSE
)
```

## Arguments

| | |
|---|---|
| `y` | output value of objective function to return |
| `id` | unique identifier for this asktell loop (default: "0") |
| `Y.tmp` | temporary "Y" values file (default: "Y.done") |
| `tmp_path` | temporary directory to store X.tmp & Y.tmp (default: 'tempdir()/../asktell.tmp') |
| `trace` | function to display asktell loop status (default : 'cat') |
| `force_cleanup` | should we cleanup temporary files before writing (possible conflicting asktell calls) ? (default: FALSE) |

## Details

'ask&tell' injection loop to call an external objective function within an inline algorithm (like optim(...)) Main idea: pass 'ask_Y' as objectve function argument of algorithm, which will wait until you call 'tell_Y' in another R process. In this secondary process, you can read what X is called using 'ask_X', and when you know what values returns from the external objective, just call 'tell_Y' to give it.

### Value

input value of objective function to compute externally

### Author(s)

Y. Richet, discussions with D. Sinoquet. Async IO principle was defined by G. Pujol.

### Examples

```
## Not run:  ### Assumes you can use two independant R sessions
## In main R session
  ask_Y(x=123)
## In another R session
  ask_X() # returns 123
  tell_Y(y=456)
## Then ask_dY in main R session returns with value '456'

## End(Not run)
```

## `to01` *Helper function to scale from [min,max] to [0,1]*

### Description

Helper function to scale from [min,max] to [0,1]

### Usage

```
to01(X, inp)
```

### Arguments

| | |
|---|---|
| `X` | values to scale |
| `inp` | list containing 'min' and 'max' values |

### Value

X scaled in [0,1]

### Examples

```
to01(10+10*data.frame(x=matrix(runif(10))),list(x=list(min=10,max=20)))
```

# Index