

# Package ‘getDTeval’

August 19, 2020

**Title** Implements Programmatically Designed Coding Statements without  
Compromising on the Runtime Performance

**Version** 0.0.1

**Depends** R (≥ 3.1.0)

**Description** The getDTeval() function facilitates the translation of the original coding statement to an optimized form for improved runtime efficiency without compromising on the programmatic coding design.  
The function can either provide a translation of the coding statement, directly evaluate the translation to return a coding result, or provide both of these outputs.

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.1

**Suggests** knitr, rmarkdown, dplyr

**Imports** data.table, formulaic, microbenchmark, stats, utils

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** David Shilane [aut],  
Anderson Nelson [ctb],  
Caffrey Lee [ctb],  
Zichen Huang [ctb],  
Mayur Bansal [ctb, cre]

**Maintainer** Mayur Bansal <mb4511@columbia.edu>

## R topics documented:

function.ending.point . . . . .	1
getDTeval . . . . .	2
translate.fn.calls . . . . .	3

---

```
function.ending.point
      function.ending.point
```

---

### Description

An Internal function to return the ending index

### Usage

```
function.ending.point(all.chars, beginning.index, ...)
```

### Arguments

<code>all.chars</code>	all the characters of the statement
<code>beginning.index</code>	specifies the index of the first character
<code>...</code>	provision for additional arguments

---

```
getDTeval      getDTeval
```

---

### Description

The getDTeval() function facilitates the translation of the original coding statement to an optimized form for improved runtime efficiency without compromising on the programmatic coding design. The function can either provide a translation of the coding statement, directly evaluate the translation to return a coding result, or provide both of these outputs

### Usage

```
getDTeval(
  the.statement,
  return.as = "result",
  coding.statements.as = "character",
  eval.type = "optimized",
  envir = .GlobalEnv,
  ...
)
```

### Arguments

<code>the.statement</code>	refers to the original coding statement which needs to be translated to an optimized form. This value may be entered as either a character value or as an expression.
<code>return.as</code>	refers to the mode of output. It could return the results as a coding statement ( <code>return.as = "code"</code> ), an evaluated coding result ( <code>return.as = "result"</code> , which is the default value), or a combination of both ( <code>return.as = "all"</code> ).

<code>coding.statements.as</code>	determines whether the coding statements provided as outputs are returned as expression objects ( <code>return.as = "expression"</code> ) or as character values ( <code>return.as = "character"</code> , which is the default).
<code>eval.type</code>	a character value stating whether the coding statement should be evaluated in its current form ( <code>eval.type = "as.is"</code> ) or have its called to <code>get()</code> and <code>eval()</code> translated ( <code>eval.type = "optimized"</code> , the default setting).
<code>envir</code>	Specify the environment for the required function. <code>.GlobalEnv</code> is set as default
<code>...</code>	provision for additional arguments

---

<code>translate.fn.calls</code>	<i>translate.fn.calls</i>
---------------------------------	---------------------------

---

## Description

Internal Function that translates programmatic designs into optimized coding statements for faster calculations

## Usage

```
translate.fn.calls(
  the.statement,
  function.name = "get(",
  envir = .GlobalEnv,
  ...
)
```

## Arguments

<code>the.statement</code>	original coding statement to perform the required calculation. Must be provided as a character value.
<code>function.name</code>	Name of the function to be translated to an optimized form. Parameter values should be either <code>'get('</code> or <code>'eval('</code> . <code>'get('</code> is set as default
<code>envir</code>	Specify the environment for the required function. <code>.GlobalEnv</code> is set as default
<code>...</code>	provision for additional arguments