Package ‘R62S3’

March 9, 2020

Title Automatic Method Generation from R6
Version 1.4.1
Description After defining an R6 class, R62S3 is used to automatically generate optional S3/S4 generics and methods for dispatch. Also allows piping for R6 objects.
Depends R (>= 3.5.0)
Imports data.table, methods
Suggests pkgdown, testthat, R6
License MIT + file LICENSE
Encoding UTF-8
LazyData true
RoxygenNote 7.0.2
BugReports https://github.com/RaphaelS1/R62S3/issues
NeedsCompilation no
Author Raphael Sonabend [aut, cre] (<https://orcid.org/0000-0001-9225-4654>)
Maintainer Raphael Sonabend <raphael.sonabend.15@ucl.ac.uk>
Repository CRAN
Date/Publication 2020-03-09 21:30:02 UTC

R topics documented:

R62Fun .................................................. 2
R62S3 ................................................... 3
R62S4 ................................................... 5
Index 8
R62Fun

Method Generator from R6 Class

Description

Auto-generates functions from an R6 Class.

Usage

R62Fun(
  R6Class,
  assignEnvir = parent.env(environment()),
  detectGeneric = TRUE,
  mask = FALSE,
  dispatchClasses = list(R6Class),
  scope = "public",
  arg1 = "object",
  exclude = NULL
)

Arguments

R6Class       R6ClassGenerator to generate public methods from
assignEnvir   environment in which to assign the generics/methods, default is parent of current
              environment.
detectGeneric logical, if TRUE (default) detects if the method has a S3 or S4 generic and
              defines functions accordingly
mask          logical, determines if non-generic functions should be masked if found, see de-
              tails.
dispatchClasses list of classes to assign dispatch methods on
scope         determines the scope of methods that should be copied, either "public", "active"
              or both
arg1          if mask == TRUE or no generic is found, then arg1 determines what name to give
              to the first argument in the generic.
exclude       an optional character vector naming the public methods or active bindings to
              exclude from the generator

Details

If scope == "public" then searches in a given R6::R6Class for all public methods that are not
initialize or clone. If scope == "active" then searches for all active bindings. Currently there
is only support for calling active bindings but not setting them. If scope == c("public", "active") then
both are included. Any methods/bindings passed to exclude will be ignored in the search.
If `mask == TRUE` then the generator ignores if a generic or method of the same name exists and will create a new function. If `mask == FALSE` then the generator will create a new generic only if an existing generic does not already exist. Methods and generics are created using standard convention. The optional `dispatchClasses` argument takes a list of `R6::R6Classes` and allows methods to be created for multiple classes at one time.

S3 generics are detected with `utils::isS3stdGeneric()` and S4 generics are detected with `methods::.S4methods()`.

### Value

Assigns generics/methods/functions to the chosen environment.

### See Also

Other R62s: `R62S3()`, `R62S4()`

### Examples

```r
printMachine <- R6::R6Class("printMachine",
    public = list(initialize = function() {},
                  printer = function(str) print(str)),
    active = list(Status = function() "Printing"))

pm <- printMachine$new()

# scope = public
R62Fun(printMachine, assignEnvir = topenv())
printer(pm, "Test String B")

# scope = active
R62Fun(printMachine, assignEnvir = topenv(), scope = "active")

# note support for accessing only, cannot assign
# values to an active binding
Status(pm)
```

---

**R62S3**  

*S3 Method Generator from R6 Class*

### Description

Auto-generates S3 generics from an R6 Class.

### Usage

```r
R62S3(
    R6Class,
    dispatchClasses = list(R6Class),
    assignEnvir = parent.env(environment()),
)```
mask = FALSE,
scope = "public",
arg1 = "object",
exclude = NULL
)

Arguments

**R6Class**
R6ClassGenerator to generate public methods from

**dispatchClasses**
list of classes to assign dispatch methods on

**assignEnvir**
environment in which to assign the generics/methods, default is parent of current environment.

**mask**
logical, determines if non-generic functions should be masked if found, see details.

**scope**
determines the scope of methods that should be copied, either "public", "active" or both

**arg1**
if mask == TRUE or no generic is found, then arg1 determines what name to give to the first argument in the generic.

**exclude**
an optional character vector naming the public methods or active bindings to exclude from the generator

Details

If scope == "public" then searches in a given R6::R6Class for all public methods that are not initialize or clone. If scope == "active" then searches for all active bindings. Currently there is only support for calling active bindings but not setting them. If scope == c("public", active") then both are included. Any methods/bindings passed to exclude will be ignored in the search.

If mask == TRUE then the generator ignores if a generic or method of the same name exists and will create a new S3 generic/method. If mask == FALSE then the generator will create a new generic only if an existing generic does not already exist. Methods and generics are created using standard convention.

The optional dispatchClasses argument takes a list of R6::R6Classes and allows methods to be created for multiple classes at one time.

S3 generics are detected with utils::isS3stdGeneric().

Value

Assigns generics/methods/functions to the chosen environment.

See Also

Other R62s: R62Fun(), R62S4()
Examples

```
printMachine <- R6::R6Class("printMachine",
    public = list(initialize = function() {},
                  printer = function(str) print(str)),
    active = list(Status = function() "Printing"))

pm <- printMachine$new()

# scope = public
R6S3(printMachine, assignEnvir = topenv())
printer(pm, "Test String B")

# scope = active
R6S3(printMachine, assignEnvir = toplevel(), scope = 'active')

# note support for accessing only, cannot assign
# values to an active binding
Status(pm)
```

---

R6S4  

**S4 Method Generator from R6 Class**

**Description**

Auto-generates S4 generics from an R6 Class.

**Usage**

```
R6S4(
    R6Class,
    dispatchClasses = list(R6Class),
    assignEnvir = parent.env(environment()),
    mask = FALSE,
    scope = "public",
    arg1 = "object",
    exclude = NULL

)
```

**Arguments**

- **R6Class**  
  R6ClassGenerator to generate public methods from
- **dispatchClasses**  
  list of classes to assign dispatch methods on
- **assignEnvir**  
  environment in which to assign the generics/methods, default is parent of current environment.
- **mask**  
  logical, determines if non-generic functions should be masked if found, see details.
scope determines the scope of methods that should be copied, either "public", "active" or both

arg1 if mask == TRUE or no generic is found, then arg1 determines what name to give to the first argument in the generic.

exclude an optional character vector naming the public methods or active bindings to exclude from the generator

Details

If scope == "public" then searches in a given R6::R6Class for all public methods that are not initialize or clone. If scope == "active" then searches for all active bindings. Currently there is only support for calling active bindings but not setting them. If scope == c("public", active") then both are included. Any methods/bindings passed to exclude will be ignored in the search.

If mask == TRUE then the generator ignores if a generic or method of the same name exists and will create a new S4 generic/method. If mask == FALSE then the generator will create a new generic only if an existing generic does not already exist. Methods and generics are created using standard convention.

The optional dispatchClasses argument takes a list of R6::R6Classes and allows methods to be created for multiple classes at one time.

S4 generics are detected with methods::.S4methods().

Value

Assigns generics/methods/functions to the chosen environment.

Assigns methods and generics to the chosen environment.

See Also

methods::setMethod methods::setGeneric

Other R62s: R62Fun(), R62S3()

Examples

```r
printMachine <- R6::R6Class("printMachine",
    public = list(initialize = function() {},
      printer = function(str) print(str),
      active = list(Status = function() "Printing"))

pm <- printMachine$new()

# scope = public
R62S4(printMachine, assignEnvir = topenv())
printer(pm, "Test String B")

# scope = active
R62S4(printMachine, assignEnvir = topenv(), scope = 'active')

# note support for accessing only, cannot assign
```
# values to an active binding
Status(pm)
Index

methods::S4methods(), 3, 6
methods::setGeneric, 6
methods::setMethod, 6

R62Fun, 2, 4, 6
R62S3, 3, 3, 6
R62S4, 3, 4, 5
R6::R6Class, 2–4, 6

utils::isS3stdGeneric(), 3, 4