

# Package: aka (via r-universe)

September 27, 2024

**Title** Define Aliases for R Expressions

**Version** 0.1.0

**Description** Create aliases for other R names or arbitrarily complex R expressions. Accessing the alias acts as-if the aliased expression were invoked instead, and continuously reflects the current value of that expression: updates to the original expression will be reflected in the alias; and updates to the alias will automatically be reflected in the original expression.

**URL** <https://klmr.me/aka/>, <https://github.com/klmr/aka>

**BugReports** <https://github.com/klmr/aka/issues>

**Suggests** testthat

**License** MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**Repository** <https://klmr.r-universe.dev>

**RemoteUrl** <https://github.com/klmr/aka>

**RemoteRef** main

**RemoteSha** d4a9c754bfc31433ab50be52444241bd93f2b158

## Contents

alias . . . . .	2
getters . . . . .	3
<b>Index</b>	<b>5</b>

---

alias *Create an alias for an expression*

---

### Description

`alias(name = expr)` creates an alias for `expr` named `name`. Subsequently, `name` can (mostly) be used interchangeably with `expr`.

`name %&=% expr` is the same as `alias(name = expr)`.

### Usage

```
alias(name = expr, expr_env = parent.frame(), alias_env = parent.frame())
```

```
name %&=% expr
```

### Arguments

<code>expr_env</code>	the environment in which to evaluate the expression
<code>alias_env</code>	the environment in which to create the alias
<code>name</code>	the alias name
<code>expr</code>	an arbitrary R expression to be aliased by <code>name</code> ; can contain interpolated expressions; see <i>Details</i>

### Details

After executing `alias(name = expr)`, `name` can be used to refer to the value of `expr`. This is especially useful when `expr` is a complex expression that is used multiple times in the code. Unlike with regular assignment, `expr` will be reevaluated every time `name` is evaluated. This means that the value of `name` always stays up to date, similar to a “reactive” expression. On the flip side, it also means that accessing `name` can be very slow if evaluating `expr` is time-consuming.

`expr` can contain interpolated expressions using the `bquote()` syntax (including splicing). These will be substituted at the time of defining the alias. See *Examples*.

The parameters `expr_env` and `alias_env` are used to control the environments in which the expression is evaluated and the alias is created, respectively. Note that specifying the correct `expr_env` is particularly important when *assigning* to an alias: an expression can be evaluated inside a parent environment without having to specify `expr_env`; however, during assignment this would cause the assignee object to be copied into the calling environment. See *Examples* for a concrete example of this.

### Value

`alias()` is called for its side-effect and does not return a value.

**Examples**

```
x = 'hello'
alias(ax = x)
ax # prints 'hello'

x = 'world'
ax # prints 'world'

ax = 'goodbye'
x # prints 'goodbye'

# Aliases can be created for complex expressions:
alias(mercedes = mtcars[grepl('^Merc ', rownames(mtcars)), ])
mercedes

mercedes$vs = 0 # set all Mercedes engine types to V-shaped
mtcars

# Aliases can contain interpolated expressions:
n = 1
m = 2
alias(s = .(n) + m)
s # prints 3

n = 10
m = 10
s # prints 11

alias_expr('s') # prints `1 + m`

# Be careful when assigning to an alias to an object in a parent environment:
e = attach(new.env())
e$y = 'hello'

alias(ay = y)

# Works: `y` is found in the parent environment
ay # prints 'hello'

# But the following creates a *new variable* `y` in the current environment:
ay = 'world'
e$y # prints 'hello', still!
y # prints 'world'

# To prevent this, use `expr_env`:
# alias(ay = y, expr_env = e)
```

**Description**

`alias_expr(alias)` returns the expression that was used to define an alias.

`alias_env(alias)` returns the environment in which the aliased expression is evaluated.

**Usage**

```
alias_expr(alias, envir = parent.frame())
```

```
alias_env(alias, envir = parent.frame())
```

**Arguments**

`alias` the name of an alias, as a string

`envir` the environment in which to look up the alias name (default: calling environment)

**Value**

`alias_expr(alias)` returns an unevaluated R expression (a name or a call).

`alias_env(alias)` returns an environment.

# Index

`%=&%` (alias), [2](#)

alias, [2](#)

alias\_env (getters), [3](#)

alias\_expr (getters), [3](#)

bquote(), [2](#)

getters, [3](#)

“reactive” expression, [2](#)