To make teaching [R](https://www.r-project.org/) [quasi-quotation](https://en.wikipedia.org/wiki/Grave_accent#Use_in_programming) easier it would be nice if R [string-interpolation](https://en.wikipedia.org/wiki/String_interpolation) and quasi-quotation both used the same notation. They are related concepts. So some commonality of notation would actually be clarifying, and help teach the concepts. We will define both of the above terms, and demonstrate the relation between the two concepts.

**String-interpolation**

[String-interpolation](https://en.wikipedia.org/wiki/String_interpolation) is the name for substituting value into a string. For example:

library("wrapr")

variable <- [as.name](http://as.name)("angle")

sinterp(

'variable name is .(variable)'

)

## [1] "variable name is angle"

Notice the ".(variable)" portion was replaced with the actual variable name "angle". For string interpolation we are intentionally using the ".()" notation that Thomas Lumley’s picked in 2003 when he introduced quasi-quotation into R (a different concept than string-interpolation, but the topic of our next section).

String interpolation is a common need, and there are many [R](https://www.r-project.org/) packages that supply variations of such functionality:

* base::sprintf
* [R.utils::gstring()](https://cran.r-project.org/package=R.utils)
* [rprintf::rprintf()](https://cran.r-project.org/package=rprintf)
* [stringr::str\_interp()](https://cran.r-project.org/package=stringr)
* [glue::glue()](https://cran.r-project.org/package=glue)
* [wrapr::sinterp()](https://winvector.github.io/wrapr/reference/sinterp.html) (requires version 1.8.3, or newer).

**Quasi-quotation**

A related idea is ["quasi-quotation"](https://en.wikipedia.org/wiki/Grave_accent#Use_in_programming) which substitutes a value into a general expression. For example:

angle = 1:10

variable <- [as.name](http://as.name)("angle")

evalb(

plot(x = .(variable),

y = sin(.(variable)))

)

Notice how in the above plot the actual variable name "angle" was substituted into the graphics::plot() arguments, allowing this name to appear on the axis labels.

evalb() is is a very simple function built on top of base::bquote():

print(evalb)

## function(..., where = parent.frame()) {

## force(where)

## exprq <- bquote(..., where = where)

## eval(exprq,

## envir = where,

## enclos = where)

## }

##

##

All evalb() does is: call bquote() as intended. A way to teach this is to just call bqoute() alone.

bquote(

plot(x = .(variable),

y = sin(.(variable)))

)

## plot(x = angle, y = sin(angle))

And we see the un-executed code with the substitutions performed.

There are many R quasi-quotation systems including:

* base::bquote()
* [gtools::strmacro()](https://cran.r-project.org/package=gtools)
* [lazyeval](https://cran.r-project.org/package=lazyeval)
* [wrapr::let()](https://winvector.github.io/wrapr/reference/let.html)
* [rlang::as\_quosure()](https://rlang.r-lib.org/reference/as_quosure.html)
* [nseval](https://cran.r-project.org/package=nseval)

If you don’t want to wrap your plot() call in evalb() you can instead pre-adapt the function. Below we create a new function plotb() that is intended as shorthand for eval(bquote(plot(...))).

plotb <- bquote\_function(graphics::plot)

plotb(x = .(variable),

y = sin(.(variable)))

**Conclusion**

When string-inerpolation and quasi-quotation use the same notation we can teach them quickly as simple related concepts.