how to pass a formula to lm().

Often when modeling in [R](https://www.r-project.org/) one wants to build up a formula outside of the modeling call. This allows the set of columns being used to be passed around as a vector of strings, and treated as data. Being able to treat controls (such as the set of variables to use) as manipulable values allows for very powerful automated modeling methods.

What we are talking about is the ability to take the outcome (or dependent variable) and modeling variables (or independent variables) from somewhere else, as data. The kind of code we are talking about is shown below.

# specifications of how to model,

# coming from somewhere else

outcome <- "mpg"

variables <- c("cyl", "disp", "hp", "carb")

# our modeling effort,

# fully parameterized!

f <- as.formula(

paste(outcome,

paste(variables, collapse = " + "),

sep = " ~ "))

print(f)

# mpg ~ cyl + disp + hp + carb

model <- lm(f, data = mtcars)

print(model)

# Call:

# lm(formula = f, data = mtcars)

#

# Coefficients:

# (Intercept) cyl disp hp carb

# 34.021595 -1.048523 -0.026906 0.009349 -0.926863

This works, and the paste() pattern is so useful we suggest researching and memorizing it.

However the “call” portion of the model is reported as “formula = f” (the name of the variable carrying the formula) instead of something more detailed. Frankly this printing issue never bothered us. None of our tools or workflows currently use the model call item, and for a very large number of variables formatting the call contents in the model report becomes unweildy. We also already have the formula in a variable, so if we need it we can save it or pass it along.

There is a much better place on many models to get model structure information from than the model call item: the model terms item. This item carries a lot of information and formats up quite nicely:

format(terms(model))

# [1] "mpg ~ cyl + disp + hp + carb"

Notice we used accessor notation (terms(model)) to get the information. List notation, such as model$terms also works.

In addition, as is so often the case in R, there is already a known solution to the above problem. For common R issues one should suspect there is a good available R solution. It is just a matter of finding the right reference or teaching. For example: to control the model$call item use the bquote() facility, as we show below.

outcome <- "mpg"

variables <- c("cyl", "disp", "hp", "carb")

f <- as.formula(

paste(outcome,

paste(variables, collapse = " + "),

sep = " ~ "))

print(f)

# mpg ~ cyl + disp + hp + carb

# The new line of code

model <- eval(bquote( lm(.(f), data = mtcars) ))

print(model)

# Call:

# lm(formula = mpg ~ cyl + disp + hp + carb, data = mtcars)

#

# Coefficients:

# (Intercept) cyl disp hp carb

# 34.021595 -1.048523 -0.026906 0.009349 -0.926863

base::bquote() is a very sensible implementation of quasi-quotation or the Lisp backquote facility. The idea is everything inside the bquote() is “quoted” (held unevaluated as an R-language tree, not as mere strings!), with the exception of anything marked with the “.()” notation. Anything marked with .() is not quoted, but substituted in by value. This is why we see the contents of our formula, and not the name of the variable we used to denote it. base::eval() is finally used to execute the combined contents.

base::bquote() has some deliberate limits (unwillingness to substitute into left-hand-sides of expressions, and some complexity of notation), which is why we promote [wrapr::let()](https://cran.r-project.org/web/packages/wrapr/vignettes/let.html) for name for name replacement tasks (wrapr::let() is for substituting a *fixed* number of symbols and combines the eval(bquote()) pattern into a single function).

In conclusion: the exact *saved* call-text in a model object may not be important, as a better structured record of the model specification is found in the model terms item. However, you can also control the model call text by evaluating the model using the eval()/bquote()/.() pattern we demonstrated above.