**Object-oriented language**

You may remember using summary()elsewhere in the notes to simply find summary measures (like the mean) for a data frame. Why did summary() work differently when using summary(mod.fit) for the GPA regression example?

Information created by functions is stored within an object. To distinguish objects that contain different collections of information, R assigns each object an attribute called a *class*. You can view them by using the attributes() or class()functions. For example,

> class(mod.fit)

[1] "lm"

> class(gpa)

[1] "data.frame"

> class(gpa$HS.GPA)

[1] "numeric"

Functions are typically designed to operate on only one or very few classes of objects. However, some functions, like summary(), are *generic*, in the sense that essentially different versions of them have been constructed to work with different classes of objects.

When a generic function is run with an object, R first checks the object's class type and then looks to find a *method* function with the name format <generic function>.<class name>. Below are examples for summary():

* summary(mod.fit) – The function summary.lm() summarizes the regression model
* summary(gpa) – The function summary.data.frame() summarizes the data frame’s contents
* summary.default() – R attempts to run this function if there is no method function for a class

There are many generic functions! For example, plot() is a generic function (try plot(mod.fit) to see what happens!). We will also see other generic functions like predict() later in the notes.

Why is R set-up like this?

The purpose of generic functions is to use a familiar language set with any object. For example, we frequently want to summarize data or a model (summary()), plot data (plot()), and find predictions (predict()), so it is convenient to use the same language set no matter the application. This is why R is referred to as an object-oriented language. The object class type determines the function action. Understanding generic functions may be one of the most difficult topics for new R users!

To see a list of all method functions associated with a class, use methods(class = <class name>). For the regression example, the method functions associated with the lm class are:

> methods(class = "lm")

 [1] add1 alias anova

 [4] case.names coerce confint

 [7] cooks.distance deviance dfbeta

[10] dfbetas drop1 dummy.coef

[13] effects extractAIC family

[16] formula hatvalues influence

[19] initialize kappa labels

[22] logLik model.frame model.matrix

[25] nobs plot predict

[28] print proj qr

[31] residuals rstandard rstudent

[34] show simulate slotsFromS3

[37] summary variable.names vcov

see '?methods' for accessing help and source code

To see a list of all method functions for a generic function, use methods(generic.function = <generic function name>). Below are the method functions associated with summary():

> methods(generic.function = "summary")

 [1] summary.aov summary.aovlist\*

 [3] summary.aspell\* summary.check\_packages\_in\_dir\*

 [5] summary.connection summary.data.frame

 [7] summary.Date summary.default

 [9] summary.ecdf\* summary.factor

[11] summary.glm summary.infl\*

[13] summary.lm summary.loess\*

[15] summary.manova summary.matrix

[17] summary.mlm\* summary.nls\*

[19] summary.packageStatus\* summary.POSIXct

[21] summary.POSIXlt summary.ppr\*

[23] summary.prcomp\* summary.princomp\*

[25] summary.proc\_time summary.sockclientconn\*

[27] summary.srcfile summary.srcref

[29] summary.stepfun summary.stl\*

[31] summary.table summary.tukeysmooth\*

[33] summary.warnings

see '?methods' for accessing help and source code

Knowing what a name of a particular method function can be helpful to find help on it. For example, the help for summary() alone is not very helpful! However, the help for summary.lm() provides a lot of useful information about what is summarized for a regression model.