Formulas in R

for ECON207, ECON233, and others

The R Formula Object

An R **formula** is an object with **mode "language"**, **type "call"**, and **class "formula"**. (We'll just say "a formula object"). It is used to specify an unevaluated expression.

The following code creates an formula object that says "y is a function of x and z" and stores it as a formula object as `fml0`. The formula object created contains three 'sub-objects', comprising the LHS expression, '~', and the RHS expression.

```
fml0 <- y ~ x + z
typeof(fml0)
```

```
## [1] "language"
```

```
mode(fml0)
```

```
## [1] "call"
```

```
class(fml0)
```

```
## [1] "formula"
```

```
length(fml0)
```

```
## [1] 3
```

```
fml0[[1]]
```

```
## `~`
```

```
fml0[[2]]
```

```
## y
```

```
fml0[[3]]
```

```
## x + z
```

The Formula Object in Regressions

How a formula object is used depends on the function. The following example uses the `mtcars` dataset and the `lm()` function to run a linear regression:

```
data(mtcars)
head(mtcars,3)
```

```
##                mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4     21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710    22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
```

```
tail(mtcars,3)
```

```
##                mpg cyl disp  hp drat   wt qsec vs am gear carb
## Ferrari Dino  19.7   6  145 175 3.62 2.77 15.5  0  1    5    6
```

```
## Maserati Bora 15.0   8  301 335 3.54 3.57 14.6  0  1    5    8
## Volvo 142E    21.4   4  121 109 4.11 2.78 18.6  1  1    4    2
```

We run a linear regression of `mpg` against `wt` and `hp`.

```
frm1 <- mpg ~ wt + hp
mdl1 <- lm(frm1, data=mtcars)
summary(mdl1)
```

```
##
## Call:
## lm(formula = frm1, data = mtcars)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.941 -1.600 -0.182  1.050  5.854
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 37.22727    1.59879  23.285  < 2e-16 ***
## wt          -3.87783    0.63273  -6.129 1.12e-06 ***
## hp          -0.03177    0.00903  -3.519  0.00145 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.593 on 29 degrees of freedom
## Multiple R-squared:  0.8268, Adjusted R-squared:  0.8148
## F-statistic: 69.21 on 2 and 29 DF,  p-value: 9.109e-12
```

The function **lm()** fits linear models. It take the formula `frm1` which says "`mpg` is a function of `wt` and `hp`" and interprets it as the linear function

$$mpg = \beta_0 + \beta_1 wt + \beta_2 hp + \epsilon$$

including the constant term. It then uses the data in the data set `mtcars` to estimate the linear regression model. After fitting the model, the function `lm()` returns a **list** object of the class **lm**. The list object includes a number of things

```
attributes(mdl1)
```

```
## $names
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
##
## $class
## [1] "lm"
```

The function **summary()** takes this list, and produces a nicely formatted regression output. If you only called `mdl1` without `summary()`, you will get a highly summarized output.

```
mdl1
```

```
##
## Call:
## lm(formula = frm1, data = mtcars)
##
## Coefficients:
```

```
## (Intercept)           wt           hp
##    37.22727     -3.87783     -0.03177
```

To save space, we'll use the short output from now. We'll also specify formulas directly in `lm()` rather than in a separate line, and print the regression results directly without storing the output as an lm object.

Before proceeding, we mention here that repeated variables on the right-hand side of a formula are ignored:

```
lm(mpg ~ wt + wt, data=mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ wt + wt, data = mtcars)
##
## Coefficients:
## (Intercept)           wt
##      37.285       -5.344
```

Formula Symbols

A number of special symbols are used in describing more complex formulas. We illustrate them in the context of the `lm()` function.

The symbol "`-`" is used for excluding terms. For instance, to exclude the intercept term, we would include "`-1`" in the formula.

```
lm(mpg ~ wt + hp - 1, data=mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ wt + hp - 1, data = mtcars)
##
## Coefficients:
##       wt        hp
##  6.84045  -0.03394
```

The symbol "`.`" is used to mean 'all variables'. To run a regression of `mpg` on all variables, but not including `carb` and not including the intercept:

```
lm(mpg ~ . - carb - 1, data=mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ . - carb - 1, data = mtcars)
##
## Coefficients:
##      cyl      disp        hp      drat        wt      qsec        vs
##  0.30057   0.01743  -0.02419   1.15898  -4.25650   1.27587   0.24519
##       am      gear
##  2.89121   0.87309
```

The symbols `:`, `*`, and `^` are also useful in specifying formulas. The latter two symbols **DO NOT** have the usual mathematical interpretations.

The symbol `:` indicates **interaction** term. In the case of two continuous variables (as opposed to factors, or categorical variables), you can think of it as their product. The following gives

$$mpg = \beta_0 + \beta_1 wt.hp + \epsilon$$

```
lm(mpg ~ wt:hp, data=mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ wt:hp, data = mtcars)
##
## Coefficients:
## (Intercept)        wt:hp
##    27.74564     -0.01487
```

The symbol * indicates 'crossing'. The formula `mpg ~ wt*hp` gives

$$mpg = \beta_0 + \beta_1 wt + \beta_2 hp + \beta_3 wt.hp + \epsilon$$

```
lm(mpg ~ wt*hp, data=mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ wt * hp, data = mtcars)
##
## Coefficients:
## (Intercept)           wt           hp        wt:hp
##    49.80842     -8.21662     -0.12010      0.02785
```

To have just wt and wt.hp, we can state `mpg ~ wt + hp:wt`:

```
lm(mpg ~ wt+hp:wt, data=mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ wt + hp:wt, data = mtcars)
##
## Coefficients:
## (Intercept)           wt        wt:hp
##    34.29735     -3.37105     -0.00653
```

Crossing three variables leads to inclusion of the variables plus all cross-products:

```
lm(mpg ~ wt*hp*disp, data=mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ wt * hp * disp, data = mtcars)
##
## Coefficients:
## (Intercept)           wt           hp         disp        wt:hp
##   5.255e+01   -7.511e+00   -1.426e-01   -6.416e-02    2.614e-02
##     wt:disp      hp:disp   wt:hp:disp
##   1.237e-02    3.297e-04   -6.295e-05
```

We can specify cross-products up to a specified degree, using ^. The following produces all cross-products of `wt`, `hp`, and `disp`, but excludes `wt:hp:disp` since crossing is limited to 2:

```
lm(mpg ~ (wt+hp+disp)^2, data=mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ (wt + hp + disp)^2, data = mtcars)
```

```
##
## Coefficients:
## (Intercept)           wt           hp         disp        wt:hp
##  48.2558247   -6.2016994   -0.1193411   -0.0198651    0.0181979
##     wt:disp      hp:disp
##  -0.0005743    0.0001238
```

You get the same result with `mpg ~ wt*hp*disp - wt:hp:disp`:

```
lm(mpg ~ wt*hp*disp - wt:hp:disp, data=mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ wt * hp * disp - wt:hp:disp, data = mtcars)
##
## Coefficients:
## (Intercept)           wt           hp         disp        wt:hp
##  48.2558247   -6.2016994   -0.1193411   -0.0198651    0.0181979
##     wt:disp      hp:disp
##  -0.0005743    0.0001238
```

We emphasize that in the context of R formulas, the `^` symbol does not indicate 'power', and and expression like `x^2` will just return `x`.

```
lm(mpg ~ wt^2, data=mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ wt^2, data = mtcars)
##
## Coefficients:
## (Intercept)           wt
##      37.285       -5.344
```

Problem: how do we specify a polynomial regression

$$mpg = \beta_0 + \beta_1 wt + \beta_2 wt^2 + \epsilon$$

If we try `mpg ~ wt + wt^2`, we'll get the simple linear regression of `mpg` on `wt` (`wt^2' evaluates towt, and as there is already a`wt', the second one is ignored.)

```
lm(mpg ~ wt + wt^2, data=mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ wt + wt^2, data = mtcars)
##
## Coefficients:
## (Intercept)           wt
##      37.285       -5.344
```

In order to accommodate such situations, R has the "as-is" function `I()`. Wrapping `x^2` in `I()` will allow it to be evaluated in the usual fashion (as a square) and be included as a separate variable.

```
lm(mpg ~ wt + I(wt^2), data=mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ wt + I(wt^2), data = mtcars)
```

```
##
## Coefficients:
## (Intercept)           wt      I(wt^2)
##      49.931      -13.380        1.171
```

Question: What specification do the following two formulas produce when used in `lm()`?

- mpg ~ (hp + wt) * drat
- mpg ~ (hp + wt) * (drat + qsec)

Factors in Formulas and Regressions

A **factor** is a categorical variable. In the data set `mtcars`, all the variables are stored as numerics

```
str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

but a number of them can be treated as factors. To illustrate factors in regressions, we convert `cyl` (no. of cylinders) into an factor variable with levels "4", "6", and "8" (the specific categories of a factor are called levels). We also convert `vs` into a factor denoting engine types 0="V", and 1="S". Note that the factor we have created is an **unordered** factor. The default behavior of `lm()` is very different for ordered factors.

```
mtcars$cyl <- factor(mtcars$cyl)
str(mtcars$cyl)
```

```
##  Factor w/ 3 levels "4","6","8": 2 2 1 2 3 2 3 1 1 2 ...
```

```
mtcars$vs <- factor(mtcars$vs, levels=c(0,1), labels=c("V", "S")) # Engine type
str(mtcars$vs)
```

```
##  Factor w/ 2 levels "V","S": 1 1 2 2 1 2 1 2 2 2 ...
```

```
mdl2 <- lm(mpg ~ cyl, data=mtcars)
mdl2
```

```
##
## Call:
## lm(formula = mpg ~ cyl, data = mtcars)
##
## Coefficients:
## (Intercept)         cyl6         cyl8
##      26.664       -6.921      -11.564
```

Dummy variables are created to indicate 6 cylinders (cyl6) and 8 cylinders (cyl8) – you don't have to do them yourself. The variable `mpg` is regressed on an intercept, and these two dummy variables. To see the **X** matrix used, we can use the function `model.matrix()`. Here we show the first 8 rows:

```
head(model.matrix(mdl2), 8)
```

```
##                   (Intercept) cyl6 cyl8
## Mazda RX4                   1    1    0
## Mazda RX4 Wag               1    1    0
## Datsun 710                  1    0    0
## Hornet 4 Drive              1    1    0
## Hornet Sportabout           1    0    1
## Valiant                     1    1    0
## Duster 360                  1    0    1
## Merc 240D                   1    0    0
```

The behavior of "*" is as expected, bearing in mind that dummy variables are created for `cyl`.

```
lm(mpg ~ cyl*hp, data=mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ cyl * hp, data = mtcars)
##
## Coefficients:
## (Intercept)         cyl6         cyl8           hp      cyl6:hp
##    35.98303    -15.30917    -17.90295     -0.11278      0.10516
##     cyl8:hp
##     0.09853
```

Why is `cyl6:cyl8` omitted?

With two factors:

```
lm(mpg ~ cyl*vs, data=mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ cyl * vs, data = mtcars)
##
## Coefficients:
## (Intercept)         cyl6         cyl8          vsS     cyl6:vsS
##      26.000       -5.433      -10.900        0.730       -2.172
##    cyl8:vsS
##          NA
```

Question: How do you interpret the parameter estimates?

Notice that we have no estimates for `cyl8:vsS`. This is because we have no observations in that category:

```
table(mtcars$cyl, mtcars$vs)
```

```
##
##      V  S
##   4  1 10
##   6  3  4
##   8 14  0
```

The following illustrates the behavior of ":" when used with factors in `lm()`:

```
lm(mpg ~ cyl:hp, data=mtcars)
```

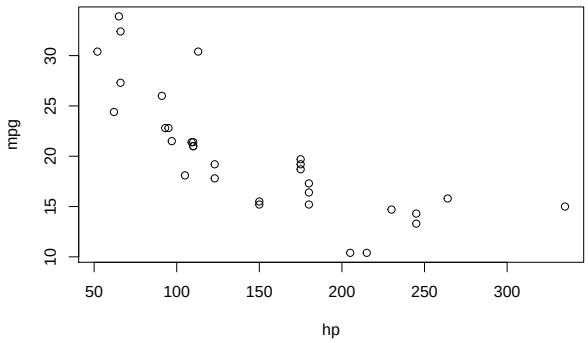
```
##
## Call:
```

```
## lm(formula = mpg ~ cyl:hp, data = mtcars)
##
## Coefficients:
## (Intercept)      cyl4:hp      cyl6:hp      cyl8:hp
##   25.389444     0.008352    -0.044917    -0.047356
```

A non-lm() application

Here are two examples of a formula being used in the `plot()` function.

```
plot(mpg~hp, data=mtcars)
```




```
par(mfrow=c(1,2))
plot(mpg~hp+wt, data=mtcars)
```