Chapter 5: Formula

Section 5.1: The basics of formula

Statistical functions in R make heavy use of the so-called Wilkinson-Rogers formula notation1 .

When running model functions like `lm` for the Linear Regressions, they need a **`formula`**. This **`formula`** specifies which regression coefficients shall be estimated.

```
my_formula1 <- formula(mpg ~ wt)
class(my_formula1)
# gives "formula"

mod1 <- lm(my_formula1, data = mtcars)
coef(mod1)
# gives (Intercept)          wt
#         37.285126   -5.344472
```

On the left side of the `~` (LHS) the dependent variable is specified, while the right hand side (RHS) contains the independent variables. Technically the **`formula`** call above is redundant because the tilde-operator is an infix function that returns an object with formula class:

```
form <- mpg ~ wt
class(form)
#[1] "formula"
```

The advantage of the **`formula`** function over `~` is that it also allows an environment for evaluation to be specified:

```
form_mt <- formula(mpg ~ wt, env = mtcars)
```

In this case, the output shows that a regression coefficient for `wt` is estimated, as well as (per default) an intercept parameter. The intercept can be excluded / forced to be 0 by including `0` or `-1` in the **`formula`**:

```
coef(lm(mpg ~ 0 + wt, data = mtcars))
coef(lm(mpg ~ wt -1, data = mtcars))
```

Interactions between variables `a` and `b` can added by included `a:b` to the **`formula`**:

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

As it is (from a statistical point of view) generally advisable not have interactions in the model without the main effects, the naive approach would be to expand the **`formula`** to `a + b + a:b`. This works but can be simplified by writing `a*b`, where the `*` operator indicates factor crossing (when between two factor columns) or multiplication when one or both of the columns are 'numeric':

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

Using the `*` notation expands a term to include all lower order effects, such that:

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

will give, in addition to the intercept, 7 regression coefficients. One for the three-way interaction, three for the two-way interactions and three for the main effects.

If one wants, for example, to exclude the three-way interaction, but retain all two-way interactions there are two shorthands. First, using `-` we can subtract any particular term:

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

Or, we can use the `^` notation to specify which level of interaction we require:

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

Those two formula specifications should create the same model matrix.

Finally, `.` is shorthand to use all available variables as main effects. In this case, the `data` argument is used to obtain the available variables (which are not on the LHS). Therefore:

```
coef(lm(mpg ~ ., data = mtcars))
```

gives coefficients for the intercept and 10 independent variables. This notation is frequently used in machine learning packages, where one would like to use all variables for prediction or classification. Note that the meaning of `.` depends on context (see e.g. `?update.formula` for a different meaning).

1. G. N. Wilkinson and C. E. Rogers. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* Vol. 22, No. 3 (1973), pp. 392-399