

Statistics 360: Advanced R for Data Science

MARS, part IV

Brad McNeney

More details on the implementation

- ▶ Input
- ▶ Pre-processing

Software arguments/inputs

- ▶ Formula interface to specify response and explanatory variables.
- ▶ data argument for input data
- ▶ object of class `mars.control` that is a list comprised of elements `Mmax`, `d` and `trace`. `Mmax` is for the forward, `d` is for the backward and `trace` is to print details of the fitting process.
 - ▶ For later (when we discuss OOP): Write a constructor, validator and helper function for this class. The helper should have defaults. If the user does not specify a `mars.control` object on input, the helper should create one with the defaults.

Pre-processing example: `lm()` vs `lm.fit()`

- ▶ `lm()` does a bit of pre-processing, including setup of the design matrix `x` and response variable `y`, and then calls `lm.fit()` to do the fitting.
- ▶ Our `fwd_selection()` and `bwd_selection()` functions will act like `lm.fit()`.
- ▶ We need code to transform the formula, data, and control parameters into inputs for `fwd_selection()`.

lm()

- `lm()` is very flexible and can read data from a data argument or the calling environment.

```
function (formula, data, subset, weights, na.action, method = "qr",
  model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,
  contrasts = NULL, offset, ...)
{
  ret.x <- x
  ret.y <- y
  cl <- match.call()
  mf <- match.call(expand.dots = FALSE)
  m <- match(c("formula", "data", "subset", "weights", "na.action",
    "offset"), names(mf), 0L)
  mf <- mf[c(1L, m)]
  mf$drop.unused.levels <- TRUE
  mf[[1L]] <- quote(stats::model.frame)
  mf <- eval(mf, parent.frame())
  if (method == "model.frame")
    return(mf)
  else if (method != "qr")
    warning(gettextf("method = '%s' is not supported. Using 'qr'",
      method), domain = NA)
  mt <- attr(mf, "terms")
  y <- model.response(mf, "numeric")
  w <- as.vector(model.weights(mf))
  if (!is.null(w) && !is.numeric(w))
    stop("'weights' must be a numeric vector")
  offset <- model.offset(mf)
  mlm <- is.matrix(y)
  ny <- if (mlm)
    nrow(y)
  else length(y)
  if (!is.null(offset)) {
    if (!mlm)
```

Pre-processing: model frames

- ▶ We'll simplify ...
- ▶ The data frame and formula are bundled into a “model frame”, which is the data frame plus a terms attribute.
- ▶ R has tools for extracting the response and design matrix from a model frame.

```
mars <- function(formula,data,control=NULL...) {  
  cc <- match.call() # save the call  
  mf <- model.frame(formula,data)  
  y <- model.response(mf)  
  mt <- attr(mf, "terms")  
  x <- model.matrix(mt, mf)  
  fwd <- fwd_stepwise(y,x,control) # Note change from earlier implement  
  bwd <- bwd_stepwise(fwd,control)  
  # Now prepare the output and return a "mars" object --  
  # to be discussed in a future lecture  
}
```