Fitting Statistical Models in Julia

Douglas Bates

06/26/2014

Statistical models and linear predictors

- Many statistical models express an *observed response vector*, $\mathbf{y}$, as a random vector, $\mathcal{Y}$, whose mean, μ, depends on a *linear predictor* expression, $\eta = \mathbf{X}\beta$.
- Elements of the *coefficient vector*, β, are parameters to be estimated from the observed data.
- $\mathbf{X}$ is the (known and fixed) *model matrix* derived from the observed values of **covariates**.
- Computational methods to estimate β and other parameters are usually described starting with $\mathbf{X}$, $\mathbf{y}$, etc.
- This whole talk is about deriving $\mathbf{X}$ from an expression, which we call the *model formula*, describing the model.
- Obtaining the form of $\mathbf{X}$ from a description can be considered the "symbolic stage" that precedes the "numeric phase" of obtaining the estimates.

Why is this not trivial?

- At a certain level you may feel that deriving **X** is just a matter of using `hcat`

```
julia> x = [1:10];
julia> y = 4.2 .+ 0.3 .* x + 0.1randn(length(x));
julia> beta = hcat(ones(length(x)),x)\y
2-element Array{Float64,1}:
 4.25619
 0.300829
```

- Indeed this is pretty close to one of the `linreg` methods

```
linreg{T<:Number}(X::StridedVecOrMat{T}, y::Vector{T})
  = [ones(T, size(X,1)) X] \ y
```

- This assumes that the only objective is to evaluate $\hat{\beta}$
- An implicit assumption is that the columns of **X** are simple to construct, say the values of numeric covariates or simple functions of them.

Stat. methods with a hidden linear model

- Those who endured an intro stat. course may have heard of
  - t-tests to compare two populations
  - paired t-tests
  - one-way, two-way, etc. analysis of variance
  - analysis of covariance
  - interaction terms
- These can all be expressed as linear least squares fits but rarely are they presented that way
- Intermediate techniques are also based on a linear predictor expression
  - logistic regression
  - Poisson regression
  - analysis of deviance
  - mixed-effects models (linear or generalized linear)
- Over half the model types described in Hastie, Tibshirani and Friedman, *The Elements of Statistical Learning* are based on linear predictor expressions.

Categorical covariates

- Categorical covariates are those that indicate membership in a group, rather than a numerical value. E.g. `subject`, `item`. The set of possible values as the *levels* of the covariate.
- In the `DataFrames` and `DataArrays` packages these are represented as a `PooledDataArray`.
- A categorical covariate generates a group of columns, derived from the indicators of the levels, in $\mathbf{X}$.
- Often the coefficients are not of interest - it's the overall "contribution" of the group of columns.
- The geometry and algebra are simple, as is the computation now but not when the techniques were formulated.
- Clever people were able to "simplify" the calculations as opaque formulas - messy but requiring fewer calculations.
- You probably learned these messy formulas that are irrelevant today. Unfortunately many people learn statistics as rote calculations.

Least squares revisited

- The basic model to be fit by least squares is $\mathcal{Y} \sim \mathcal{N}(\mathbf{X}\beta, \sigma^2\mathbf{I_n})$.
- For this model contours of constant probability are spheres centered at $\mathbf{X}\beta$. The *maximum likelihood estimates*, $\hat{\beta}$, satisfy

$$\hat{\beta} = \arg\min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|^2$$

- A direct method for determining these estimates uses a QR decomposition, which we will write as $\mathbf{X} = \mathbf{QR}$ where $\mathbf{Q}$ is the same size as $\mathbf{X}$ and satisfies $\mathbf{Q}'\mathbf{Q} = \mathbf{I_p}$ and $\mathbf{R}$ is $p \times p$ upper triangular. (Statisticians denote the number of observations as n and the number of parameters as p so $\mathbf{X}$ is $n \times p$).
- $\hat{\beta}$ satisfies $\mathbf{R}\widehat{\beta} = \mathbf{Q}'\mathbf{y}$. The vector $\mathbf{Q}'\mathbf{y}$ is called the *effects* vector.
- **If X has full column rank** then $\mathbf{R}$ is a non-singular upper triangular matrix and $\hat{\beta}$ is easily determined by backsolving.

It's the effects not the coefficients.

- For categorical covariates, we are often more interested its section of $\mathbf{Q}'\mathbf{y}$ than $\hat{\beta}$. These are the *main effects* for the term. Interaction terms also generate groups of columns and corresponding *interaction effects*.
- The squared lengths of these sections are the (sequential) *sums of squares* for these terms.
- The number of elements in the section is the *degrees of freedom* (dimension of a linear subspace) for the term.
- The simple way of generating a group of columns for a factor is to use the indicators of the levels. The sum of these columns is a column of ones.
- If we have more than 1 such term or if we have a constant term (a column of 1's) in the model, including all the indicators results in a rank deficiency.

Dealing with rank deficiency

- We can include all the columns in $\mathbf{X}$, check for numerical singularity and adjust.
- This is the "burning the toast and then scraping it" approach
- If we do the symbolic analysis first, we can avoid the rank deficiency by using "contrasts" instead of indicators.
- For a factor with k levels, a set of contrasts is any $k-1$ linear combinations, `C`, such that `[ones(n) C]` has full rank.
- Sometimes we just want a set of contrasts so we start with the full set of indicators and drop one. For definiteness we drop the first one resulting in the *treatment contrasts*.
- Other times we may choose an orthogonal set of contrasts for numerical stability, or a particular set of contrasts that provides convenient interpretation of the coefficients.
- Another approach is to include all the indicators and add a regularization or penalty term in the objective.

Interaction terms and hierarchy

- To model a situation in which the effect of one term depends on the level of another term we use *interaction terms*.
- The *order* of an interaction is the number of terms in the interaction. By extension the order of a main-effects term is 1 and the order of the constant term is 0.
- With few exceptions, sensible models obey the *heredity principle* that an interaction term should follow any lower-order terms contained in it. This is because the decomposition into "effects" is sequential.
- To achieve this we sort terms by their order.
- Often the sequence of terms of the same order is important so we should use a stable sort.
- For screening purposes we may want a model that includes all possible interactions.

The formula language (finally!)

- For this description `y` is the response, `f`, `g`, `h`, `...` are categorical covariates and `u,v,w,...` are numeric covariates.
- The constant term is implicit. It can be written explicitly as `1`. `0` suppresses the constant term.
- Interaction terms are written `f&g`, etc.
- Crossed factors are written `f*g` which expands to `f + g + f&g`

```
y ~ 1 + u    # simple linear regression
y ~ u        # same (1 is implicit)
y ~ 0 + u    # suppress intercept
y ~ 1 + f    # constant and k-1 contrasts (1-way anova)
y ~ 0 + f    # all k indicators from f
y ~ f + g    # two-way anova
y ~ f*g      # two-way anova with interaction
y ~ f + u    # parallel lines for levels of f
y ~ f * u    # non-parallel lines for levels of f
```

Interaction terms and random-effects terms

- Interaction terms of the form `f&g` are always constructed from contrasts and consist of all possible (element-wise) products of columns of indicators. If `f` has `k` levels and `g` has `l` levels then `f&g` generates $(k-1)(l-1)$ columns.
- An interaction term of the form `u&f` is the single column for `u` multiplied by each of the `k-1` contrasts for `f`.
- When the number of levels of a factor is large and these levels are considered as a sample from a population we often switch to the regularization approach using *random-effects* terms. (Terms that are not random-effects terms are called *fixed-effects* terms.)
- A simple, scalar random effects term, written `(1|h)` generates a full set of `m` indicators as a sparse matrix plus a penalty expression.
- A "random intercept" term, written `(u|h)` generates 2`m` columns consisting of all products of the indicators for `h` and the model matrix for `1 + u` plus the penalty expression.
- In general the expression on the left-hand side of the `|` is

From Formula and DataFrame to ModelFrame.

Rather than go directly from Formula and DataFrame to ModelMatrix we first evaluate a `ModelFrame` 1. Given a formula - expand terms of the form $f * g * \ldots$ to main effects and interactions - sort the terms according to their order, retaining the original sequence for terms of the same order - scan the model terms for *eterms* (evaluation terms) which are symbols, numbers or function calls that are not in the special operators.

2. From the evaluation terms and the DataFrame
   - reduce the data frame to the variables needed for the response and the evaluation terms
   - check for missing values in these variables and perform the desired *NA-action* (usually case-wise deletion). Record the pattern of deletions (bitarray)
   - evaluate the *eterms* to create the ModelFrame.

Evaluate the ModelMatrix

- separate the fixed-effects and random-effects terms
- for the fixed-effects terms
  - create the column blocks associated with the terms (sorted by increasing order)
  - if the constant term is not present the first main-effects term should generate indicators not contrasts
  - use `hcat` to generate $\mathbf{X}$ and a vector of indices of columns to terms.
- for the random-effects terms
  - evaluate the lhs as a model matrix.
  - generate the sparse interactions.