Lab 4: Least Squares Regression

Stat 154, Spring 2018

The purpose of this lab is to review some computational aspects of ordinary least squares (OLS) regression, as well as some practical implications.

Linear Models and OLS solution

Consider a response variable Y, and $X_1, \dots, X_p$ predictors observed on n individuals. For convenience purposes, assume that $n > p$.

Assume a linear model:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p + \varepsilon$$

where the error term ε is assumed to have a constant variance σ^2.

Using vector-matrix notation, we express the above model as:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

where:

- $\mathbf{y}$ is an n-vector of response values.
- $\mathbf{X}$ is an $n \times (p+1)$ matrix with a predictor variable in each column.
- the first column of $\mathbf{X}$ is a vector of ones.
- $\boldsymbol{\beta}$ is a $p+1$-vector of regression coefficients.
- $\boldsymbol{\varepsilon}$ is an n-vector of error values.

If the matrix $\mathbf{X}$ is of full column-rank—i.e. $rank(\mathbf{X}) = p+1$—then the least squares estimates $\hat{\boldsymbol{\beta}}$ can be obtained as:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$$

and the OLS predicted (or fitted) response values $\hat{\mathbf{y}}$ are given by:

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$$

Common Outputs in OLS

From a computational standpoint, OLS procedures typically involve calculating the following primary outputs:

- coefficient estimates: $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$
- fitted values: $\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n$
- residuals: $e_i = y_i - \hat{y}_i$

Secondary results are directly derived from the calculated residuals:

- Residual Sum of Squares (RSS):

$$\text{RSS} = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- Variance unbiased estimator: $\hat{\sigma}^2$

$$\hat{\sigma}^2 = \frac{\text{RSS}}{n - p - 1}$$

Using RSS, a sum of squares decomposition, as well as the so-called coefficient of determination R^2, are also standard regression outputs:

$$\text{TSS} = \text{ESS} + \text{RSS}$$

where:

- TSS is the *Total Sum of Squares*:

$$\text{TSS} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

- ESS is the *Explained Sum of Squares*:

$$\text{ESS} = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2$$

- Coefficient of determination:

$$R^2 = \frac{\text{ESS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}} = cor^2(\mathbf{y}, \hat{\mathbf{y}})$$

Part 1: Exploratory Data Analysis (EDA)

You will have to compute the OLS outputs using a handful of variables from the data set `mtcars`:

$$\texttt{mpg} = \beta_0 + \beta_1 \texttt{hp} + \beta_2 \texttt{qsec} + \beta_3 \texttt{wt} + \epsilon$$

where:

- `mpg`: miles per gallon
- `hp`: gross horsepower
- `qsec`: 1/4 mile time
- `wt`: weight (1000 lbs)

Before writing code to compute the OLS outputs, spend some time exploring all the variables listed above (i.e. response and predictors)

- summary statistics
- graphs of distributions (e.g. histograms, boxplots)
- matrix of correlation of all variables (i.e. response and predictors)
- scatterplot matrix
- principal components analysis (of all variables)

Part 2: OLS Outputs

Your mission is to write R code to define and calculate—without using the `lm()` function—the objects listed below. Ideally (although this is not mandatory), you should try to encapsulate your code by writing a couple of functions; this is a good programming habit that will make your code more reusable.

- `X`: $n \times (p+1)$ matrix
- `y`: n-vector of response values
- `coefficients`: vector of coefficients estimates
- `fitted_values`: vector of fitted values
- `residuals`: vector of residuals
- `RSS`: residual sum of squares
- `sigma2`: unbiased estimator for the variance
- `TSS`: total sum of squares
- `ESS`: explained sum of squares
- `R2`: coefficient of determination
- verify that $R^2 = cor^2(\mathbf{y}, \hat{\mathbf{y}})$

Here's a reminder of some operators and functions for matrices that you can use to obtain the required outputs:

- `as.matrix()`: convert/coerce an R object to a `"matrix"` object
- `cbind()`: column binding (to combine objects by columns)
- `rbind()`: row binding (to combine objects by rows)
- `%*%` is the "standard" matrix product operator
- `t()` computes the transpose
- `solve()` can be used to obtain the inverse of an invertible matrix

Here's the code with `lm()` so you can check and test your computations against those provided by R:

```
# regression output with lm()
reg <- lm(mpg ~ hp + qsec + wt, data = mtcars)
names(reg)
```

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

```
reg
```

```
##
## Call:
## lm(formula = mpg ~ hp + qsec + wt, data = mtcars)
##
## Coefficients:
## (Intercept)           hp         qsec           wt
##    27.61053     -0.01782      0.51083     -4.35880
```

Part 3: QR Decomposition

Say you have an R vector `y` and an R matrix `X` for $\mathbf{y}$ and $\mathbf{X}$, respectively. The OLS estimates, via the so-called normal equations, involve calculating the inverse of $\mathbf{X}^\mathsf{T}\mathbf{X}$. In R, the estimated coefficients $\hat{\beta}$ could be computed with the following code:

```
# regression coefficients
b <- solve(t(X) %*% X) %*% t(X) %*% y
```

While the previous code may return the correct answer, obtaining coefficients by directly calculating $(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}$ is not recommended. Why? Because of numerical problems that were discovered early after the introduction of computers.

As an alternative, in the beginning of the 1960s so-called *orthogonalilzation methods* were proposed for the solution of linear least squares problems. While these proposals are still not perfect, they have proven to be more stable.

The most common orthogonaliztion method is the **QR decomposition**. In this decomposition, any matrix $\mathbf{X}$ can be written as:

$$\mathbf{X} = \mathbf{QR}$$

where:

- $\mathbf{Q}$ is an $n \times p$ matrix with orthonormal columns: $\mathbf{Q}^\mathsf{T}\mathbf{Q} = \mathbf{Q}\mathbf{Q}^\mathsf{T} = \mathbf{I}$
- $\mathbf{R}$ is a $p \times p$ nonsingular upper triangular matrix

We can use the QR factorization to compute the least squares approximate solution. To see how, let $\mathbf{X} = \mathbf{QR}$ be the QR factorization of $\mathbf{X}$. We can reexpress the normal equations $\mathbf{X}^\mathsf{T}\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}^\mathsf{T}\mathbf{y}$ in terms of $\mathbf{Q}$ and $\mathbf{R}$ as:

$$\mathbf{R}^\mathsf{T}\mathbf{Q}^\mathsf{T}\mathbf{Q}\mathbf{R}\hat{\boldsymbol{\beta}} = \mathbf{R}^\mathsf{T}\mathbf{Q}^\mathsf{T}\mathbf{y}$$

which can be simplified to:

$$\mathbf{R}\hat{\boldsymbol{\beta}} = \mathbf{Q}^\mathsf{T}\mathbf{y}$$

Thus, we could obtain estimates given by:

$$\hat{\boldsymbol{\beta}} = \mathbf{R}^{-1}\mathbf{Q}^\mathsf{T}\mathbf{y}$$

In turns out that we don't actually need to compute $\mathbf{R}^{-1}$. Instead, we compute $\mathbf{R}^{-1}\mathbf{Q}^\mathsf{T}\mathbf{y}$ using **back substitution**.

Here is the algorithm to find $\hat{\boldsymbol{\beta}}$ via QR decomposition:

1. QR factorization: compute the QR decomposition $\mathbf{X} = \mathbf{QR}$
2. Compute $\mathbf{Q}^\mathsf{T}\mathbf{y}$
3. Back substitution: solve the triangular equation $\mathbf{R}\hat{\boldsymbol{\beta}} = \mathbf{Q}^\mathsf{T}\mathbf{y}$ (i.e. to find $\hat{\boldsymbol{\beta}}$)

QR factorization in R

R provides a set of functions that allow you compute the QR decomposition, and related results, of a matrix.

- `qr()`: computes the QR decomposition of a matrix
- `qr.Q()`: returns the $\mathbf{Q}$ matrix
- `qr.R()`: returns the $\mathbf{R}$ matrix
- `backsolve()`: solves a triangular system of linear equations.

Your Turn

Write an R function `qr_ols()` that takes a matrix `X` and a vector `y`, and returns a vector of coefficients obtained via the QR decomposition. You should be able to call `qr_ols()` as:

```
b <- qr_ols(X, y)
```

You can check your code against the output of `qr.solve()`

```
qr.solve(X, y)
```

Part 4

Most textbooks provide descriptions and examples of OLS solutions using "raw" data. In practice, however, analysts often fit models using transformed data. The most common transformations being mean-centered data, and standardized data.

Mean-centered predictors

- Use your `qr_ols()` function to recompute OLS coefficients by regressing untransformed `mpg` on mean-centered predictors `hp`, `qsec`, `wt`.
- From the obtained estimates, how can you recover the estimates of the untransformed predictors?

Standardized predictors

- Use your `qr_ols()` function to recompute OLS coefficients by regressing untransformed `mpg` on standardized predictors `hp`, `qsec`, `wt`.
- From the obtained estimates, how can you recover the estimates of the untransformed predictors?

Part 5: Handling Categorical Variables

Not all predictors are quantitative variables. Sometimes you will also have to deal with some categorical predictors. In OLS regression, the typical treatment for a categorical variable is to transform it into a set of dummy indicators. For example, say you have a categorical predictor $\mathbf{x}$ with $k = 3$ categories a, b, and c. The way $\mathbf{x}$ would get *dummified* for an OLS regression model typically involves creating $k - 1 = 2$ dummy indicators.

Here's a toy example with a vector $\mathbf{x} = (a, b, c, b, c, a)$, in which category a becomes the reference category, and then dummy indicators for b and c are generated:

$$\mathbf{x} = \begin{pmatrix} a \\ b \\ c \\ b \\ c \\ a \end{pmatrix} \quad \Longrightarrow \quad \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Your turn:

- Consider the variable `gear` (number of forward gears). Let's handle this variable as a categorical one.
- Form a new design matrix $\mathbf{X}$ by adding two dummy indicators for number of gears `4` and `5`.
- Calculate a new set of coefficient estimates with these new set of predictors.
- Compare your results with those returned by the following call of `lm()`

```r
# regression output with lm()
lm(mpg ~ hp + qsec + wt + factor(gear), data = mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ hp + qsec + wt + factor(gear), data = mtcars)
##
## Coefficients:
##   (Intercept)             hp           qsec             wt  factor(gear)4
##      24.13327       -0.02164        0.56761       -3.66248        1.15593
## factor(gear)5
##       2.24468
```