Lecture 2: Linear methods for regression

Rafael A. Irizarry and Hector Corrada Bravo

January, 2010

The next three lectures will cover basic methods for regression and classification. We'll see linear methods and tree-based for both in some detail, and will see nearest-neighbor methods without details for comparison. Regardless of lecture title, they are pretty much one long sequence on basics seen through linear methods and treee-based methods.

Terminology and notation

We will be mixing the terminology of statistics and computer science. For example, we will sometimes call Y and X the outcome/predictors, sometimes observed/covariates, and even input/output.

We will denote predictors with X and outcomes with Y (quantitative) and G (qualitative). Notice G are not numbers, so we cannot add or multiply them.

Height and weight are *quantitative measurements*. These are sometimes called continuous measurements.

Gender is a *qualitative measurement*. They are also called categorical or discrete. This is a particularly simple example because there are only two values. With two values we sometimes call it *binary*. We will use G to denote the set of possible values. For gender it would be $G = \{Male, Female\}$. A special case of qualitative variables are *ordered qualitative* where one can impose an order. With men/women this can't be done, but with, say, $G = \{low, medium, high\}$ it can.

For both types of variables, it makes sense to use the inputs to predict the output. Statisticians call the prediction task *regression* when the outcome is quantitative and *classification* when we predict qualitative outcomes. We will see that these have a lot in common and that they can be viewed as a task of function approximation (as with scatter plots).

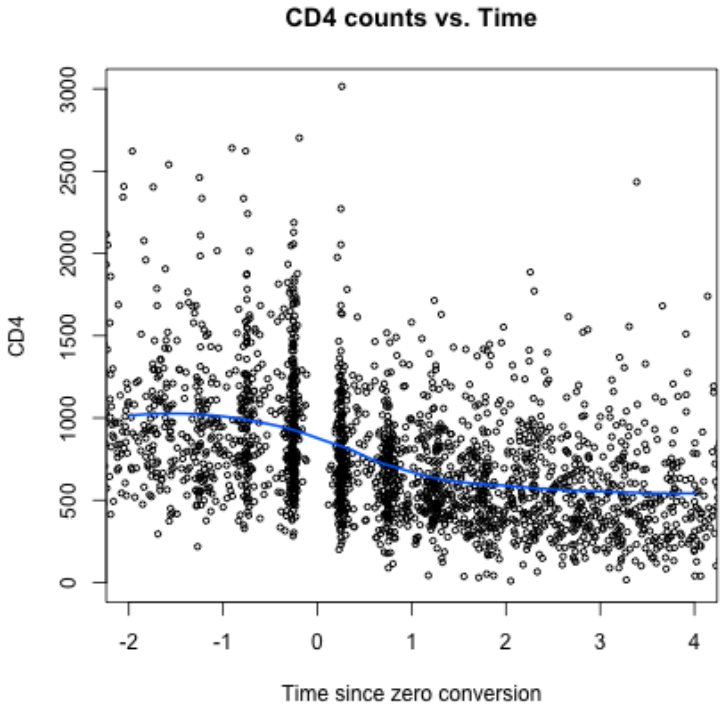
Notice that inputs also vary in measurement type.

Technical notation

We will follow the notation of Hastie, Tibshirani and Friedman. Observed values will be denoted in lower case. So x_i means the ith observation of the random variable X. Matrices are represented with bold face upper case. For example $\mathbf{X}$ will represent all observed predictors. N will usually mean the number of observations, or length of Y. i will be used to denote which observation and j to denote which covariate or predictor. Vectors will not be bold, for example x_i may mean all predictors for subject i, unles it is the vector of a particular predictor $\mathbf{x}_j$. All vectors are assumed to be column vectors, so the i-th row of $\mathbf{X}$ will be x_i', i.e., the transpose of x_i.

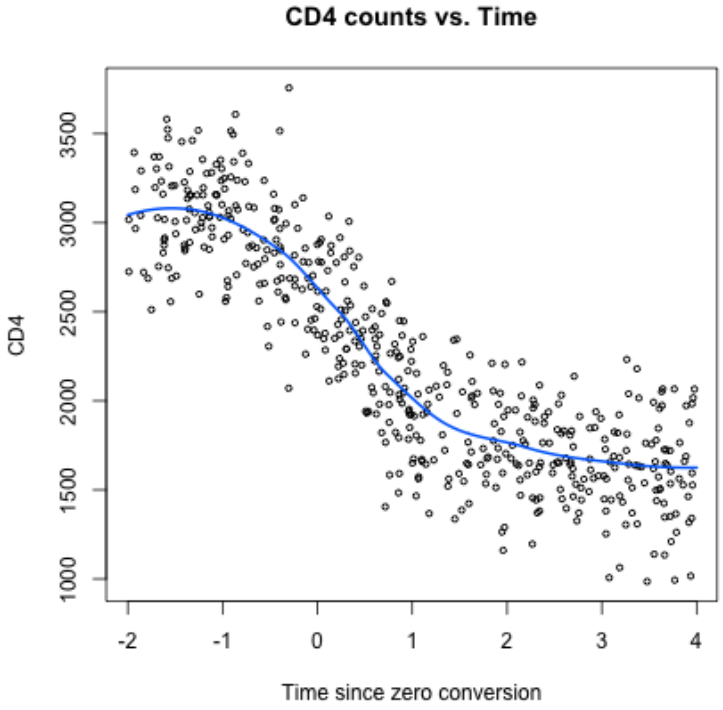
A regression problem

Recall the example data from AIDS research mentioned previously. Here we are plotting the data along with a curve from which data could have plausibly been generated. In fact, this curve was estimated from the data by a regression technique called *loess*, which we will discuss in a future lecture.



For now, let's consider this curve as truth and simulate CD4 counts from it. We

will use this simulated data to compare two simple but commonly used methods to predict Y (CD4 counts) from X (Time), and discuss some of the issues that will arise throughout this course. In particular, what is overfitting, and what is the bias-variance tradeoff.



Linear regression

Probably the most used method in statistics. In this case, we predict the output Y via the model

$$Y = \beta_0 + \beta_1 X.$$

However, we do not know what β_0 or β_1 are.

We use the training data to *estimate* them. We can also say we train the model on the data to get numeric coefficients. We will use the hat to denote the estimates: $\hat{\beta}_0$ and $\hat{\beta}_1$.

We will start using β to denote the vector $(\beta_0, \beta_1)'$. A statistician would call these the parameters of the model.

The most common way to estimates βs is by least squares. In this case, we choose the β that minimizes

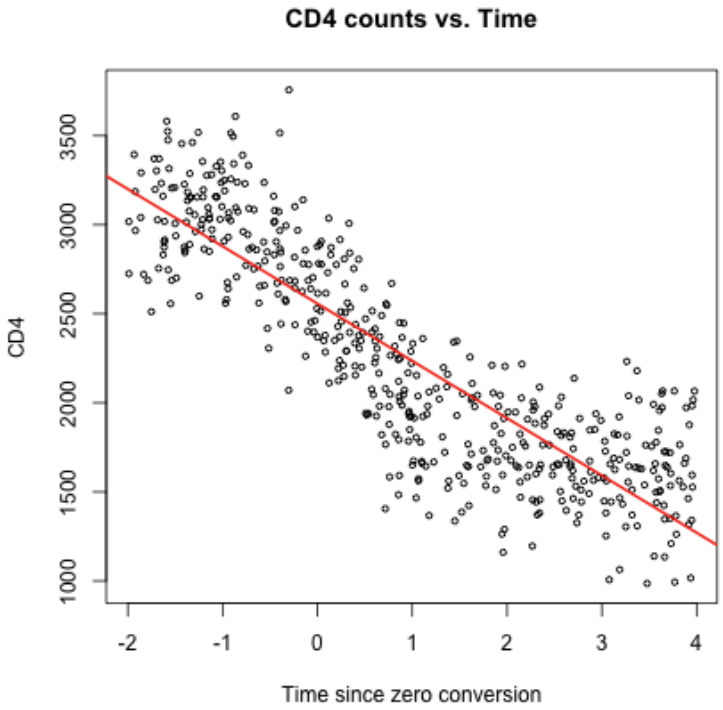
$$RSS(\beta) = \sum_{i=1}^{N} \{y_i - (\beta_0 + \beta_1 X_i)\}^2.$$

If you know linear algebra and calculus you can show that $\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}X'y$.

Notice we can predict Y for any X:

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X$$

The next Figure shows the prediction graphically. However, the data seems to suggest we could do better by considering more flexible models.



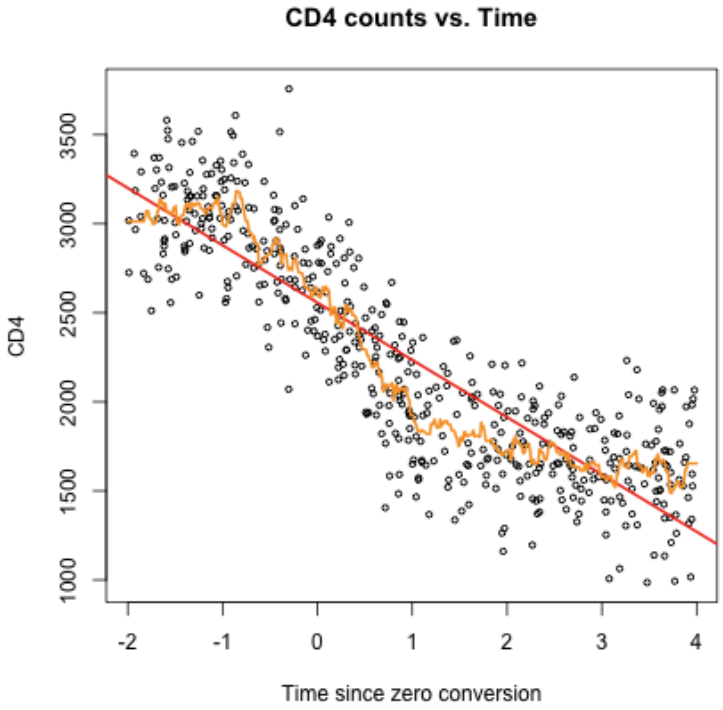
K-nearest neighbor

Nearest neighbor methods use the points closest in predictor space to x to obtain an estimate of Y. For the K-nearest neighbor method (KNN) we define

$$\hat{Y} = \frac{1}{k} \sum_{x_k \in N_k(x)} y_k.$$

Here $N_k(x)$ contains the k-nearest points to x. Notice, as for linear regression, we can predict Y for any X.

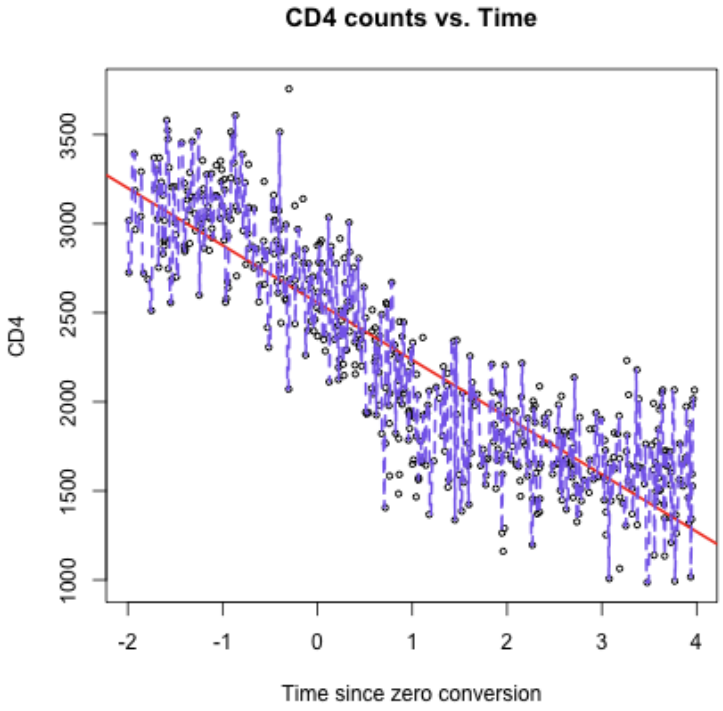
In the next Figure we see the results of KNN using the 15 nearest neighbors. This estimate looks better than the linear model.



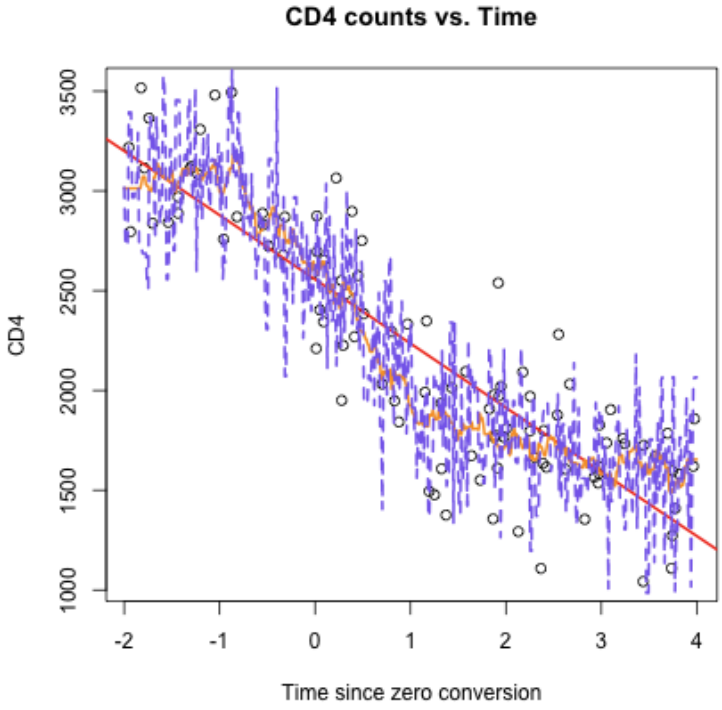
We do better with KNN than with linear regression. However, we have to be careful about *overfitting*.

Roughly speaking, *overfitting* is when you mold an algorithm to work very well (sometimes perfect) on a particular data set forgetting that it is the outcome of a random process and our trained algorithm may not do as well in other instances.

Next, we see what happens when we use KNN with k=1. In this case we make no mistakes in prediction, but do we really believe we can do well in general with this estimate?


CD4 counts vs. Time
CD4
1000
1500
2000
2500
3000
3500
-2
-1
0
1
2
3
4
Time since zero conversion

It turns out we have been hiding a *test* data set. Now we can see which of these trained algorithms performs best on an independent test set generated by the same stochastic process.



We can see how good our predictions are using RSS again.

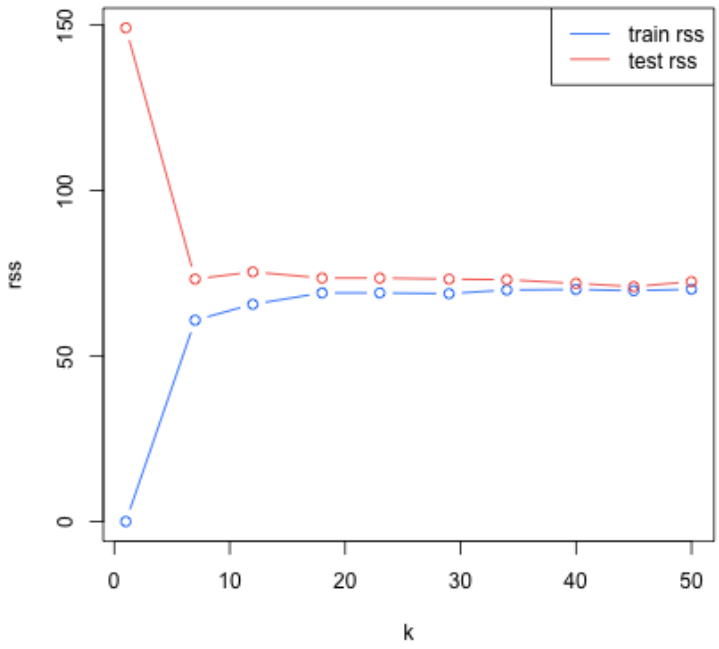
Method	Train set	Test set
Linear	99.70	93.58
K=15	67.41	75.32
K=1	0.00	149.10

Notice RSS is worse in test set than in the training set for the KNN methods. Especially for KNN=1. The spikes we had in the estimate to predict the training data perfectly no longer helps.

So, how do we choose k? We will study various ways. First, let's talk about the bias/variance trade-off.

Smaller k give more flexible estimates, but too much flexibility can result in over-fitting and thus estimates with more variance. Larger k will give more stable estimates but may not be flexible enough. Not being flexible is related to being biased.

The next figure shows the RSS in the test and training sets for KNN with varying k. Notice that for small k we are clearly overfitting.



An illustration of the bias-variance tradeoff

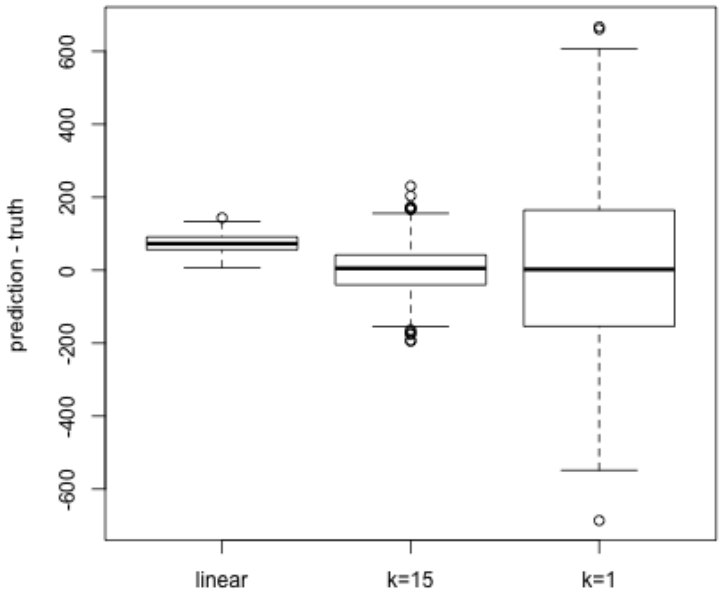
The next figure illustrates the bias/variance tradeoff. Here we plot histograms of $f(1) - \hat{f}(1)$, where $\hat{f}(1)$ is the estimate for each method trained on 1000 simulations.

We can see that the prediction from the linear model is consistently inaccurate. That is, it is biased, but stable (little variance). For $k = 1$ we get the opposite, there is a lot of variability, but once in a while it is very accurate (unbiased). For $k = 15$ we get a decent tradeoff of the two.

In this case we have simulated data as follows: for a given x

$$Y = f(x) + \epsilon$$

where $f(x)$ is the "true" curve we are using and ϵ is normal with mean zero and some variance σ^2.


prediction - truth
600
400
200
0
-200
-400
-600
linear
k=15
k=1

We have been measuring how good our predictions are by using RSS. Recall from the last lecture that we sometimes refer to this as a *loss function*. Recall also that for this loss function, if we want to minimize the **expected prediction error** for a given x:

$$E_{Y|X=x}[\{Y - f(X)\}^2|X = x],$$

we get the conditional expectation $f(x) = E[Y|X = x]$. With some algebra we see that the RSS for this optimal selection is σ^2 in our setting. That is, we can't do better than this, on average, with any other predictor.

Notice that KNN is an intuitive estimator of this optimal predictor. We do not know the function $E[Y|X = x]$ looks like so we estimate it with the y's of nearby x's. The larger k is, the less precise my estimate might be since the radius of x's I use for is larger.

Predictions are not always perfect.

Linear methods for regression

Linear predictors

Before computers became fast, linear regression was almost the only way of attacking certain prediction problems. To see why, consider a model such as this

$$Y = \beta_0 + \beta_1 e^{\beta_2 X} + \epsilon$$

finding the βs that minimize, for example, least squares is not straight forward. A grid search would require many computations because we are minimizing over a 3-dimensional space.

Technical note: For minimizing least squares in this case the Newton-Raphson algorithm would work rather well. But we still don't get an answer in closed form.

As mentioned, the least squares solution to the linear regression model:

$$Y = \beta_0 + \sum_{j=1}^{p} \beta_j X_j + \epsilon$$

has a closed form *linear solution*. In Linear Algebra notation we write: $\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'y$, with $\beta = (\beta_0, \beta_1, \ldots, \beta_j)'$. The important point here is that for any set of predictors x the prediction can be written as a linear combination

of the observed data $\hat{Y} = \sum_{i=1}^{N} w_i(x) y_i$. The $w_i(x)$ are determined by the X_js and do not depend on $\mathbf{y}$.

What is the prediction $\hat{Y}$ for x.

When we say linear regression we do not necessarily mean that we model the Y as an actual line. All we mean is that the expected value of Y is a linear combination of predictors. For example, this is a linear model:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$$

To see this, simply define $X_1 = X$, $X_2 = X^2$ and $X_3 = X^3$.

For the model we saw above, we cannot do the same because $X_1 = e^{\beta_2} X$ contains a parameter.

If the linear regression model holds, then the least squares solution has various nice properties. For example, if the ϵs are normally distributed, then $\hat{\beta}$ is the maximum likelihood estimate and is normally distributed as well. Estimating the variance components is simple: $(\mathbf{X}'\mathbf{X})^{-1}\sigma^2$ with σ^2 the error variance $\mathrm{var}(\epsilon)$. σ^2 is usually well estimated using the residual sum of squares.

If the ϵs are independent and identically distributed (IID), then $\hat{\beta}$ is the linear unbiased estimate with the smallest variance. This is called the Gauss-Markov theorem.

Technical note: Linear regression also has a nice geometrical interpretation. The prediction is the orthogonal projection of the vector defined by the data to the *hyper-plane* defined by the regression model. We also see that the least squares estimates can be obtained by using the Gram-Schmidt algorithm, which orthogonalizes the covariates and then uses simple projections. This algorithm also helps us understand the QR decomposition. For more details see Hastie, Tibshirani and Friedman.

Testing hypotheses

The fact that we can get variance estimates from regression, permits us to test simple hypotheses. For example,

$$\frac{\hat{\beta}_j}{\mathrm{se}(\hat{\beta}_j)} \sim t_{N-p-1}$$

under the assumption of normality for ϵ. When ϵ is not normal but IID, then the above is asymptotically normal.

If we want to test significance of various coefficients, we can generalize to the F-test:

$$\frac{(RSS_0 - RSS_1)/(p_1 - p_0)}{RSS_1/(N - p_1 - 1)}$$

RSS_1 is for the least squares fit of the bigger model with p_1+1 parameters, and RSS_0 is the same for the smaller model with p_0+1 parameters, having p_1-p_0 parameters constrained to be 0.

Under normality assumptions this statistic (the F-statistic) follows a $F_{p_1-p_0,N-p_0-p_1}$ distribution.

Similarly, we can form confidence intervals (or balls). For the case of multiple coefficients, we can use the fact that

$$\frac{(\hat{\beta}-\beta)'\mathbf{X}'\mathbf{X}(\hat{\beta}-\beta)}{\hat{\sigma}^2}$$

follows a χ^2_{p+1} distribution.

Gram-Schmidt

One can show that the regression coefficient for the j-th predictor is the simple regression coefficient of y on this predictor adjusted for all others (obtained using Gram-Schmidt).

For the simple regression problem (with no intercept)

$$Y = X\beta + \epsilon$$

the least square estimate is

$$\hat{\beta} = \frac{\sum_{i=1}^N x_i y_i}{\sum_{i=1}^N x_i^2}$$

Can you see for the constant model?

Mathematicians write the above solution as

$$\hat{\beta} = \frac{\langle x, y\rangle}{\langle x, x\rangle}$$

We will call this operation regressing $\mathbf{y}$ on $\mathbf{x}$ (it's the projection of $\mathbf{y}$ onto the space spanned by $\mathbf{x}$).

The residual can the be written as

$$\mathbf{r} = \mathbf{y} - \beta\mathbf{x}$$

What was the solution for β_1 to $Y = \beta_0 + \beta_1 X + \epsilon$?

We can write the result as

$$\hat{\beta}_1 = \frac{\langle \mathbf{x} - \bar{x}\mathbf{1}, \mathbf{y} \rangle}{\langle \mathbf{x} - \bar{x}\mathbf{1}, \mathbf{x} - \bar{x}\mathbf{1} \rangle}$$

Regression by Successive Orthogonalization

1. Initialize $\mathbf{z_0} = \mathbf{x_0} = \mathbf{1}$
2. For $j = 1, 2, \ldots, p$

   Regress $\mathbf{x_j}$ on $\mathbf{z_0}, \mathbf{z_1}, \ldots, \mathbf{z_{j-1}}$ to produce coefficients $\gamma_{lj} = \frac{\langle \mathbf{z_l}, \mathbf{x_j} \rangle}{\langle \mathbf{z_l}, \mathbf{z_l} \rangle}$, $l = 0, \ldots, j-1$ and residual vector $\mathbf{z_j} = \mathbf{x_j} - \sum_{k=0}^{j-1} \gamma_{kj} \mathbf{z_k}$
3. Regress y on the residual $\mathbf{z_p}$ to give estimate $\hat{\beta}_p$

Notice that the Gram-Schmidt algorithm permits us to estimate the β_j in a multivariate regression problem by successively regressing (orthogonalizing) $\mathbf{x}_j$ to produce residual vectors that form an orthogonal basis for the column space of $\mathbf{X}$. The least squares estimate is found by regressing $\mathbf{y}$ on the final residuals, i.e. projecting on this orthogonal basis.

Notice that if all the $\mathbf{x}$'s are correlated then each predictor affects the coefficients of the others.

The interpretation is that the coefficient of a predictor $\hat{\beta}_j$ is the regression of $\mathbf{y}$ on $\mathbf{x}_j$ after $\mathbf{x}_j$ has been *adjusted* for all other predictors.