Lecture 17: Multicollinearity

36-401, Fall 2015, Section B

27 October 2015

Contents

1 Why Collinearity Is a Problem **1**
1.1 Dealing with Collinearity by Deleting Variables 2
1.2 Diagnosing Collinearity Among Pairs of Variables 3
1.3 Why Multicollinearity Is Harder 3
1.4 Geometric Perspective 5

2 Variance Inflation Factors **5**
2.1 Why $VIF_i \geq 1$ 5

3 Matrix-Geometric Perspective on Multicollinearity **7**
3.1 The Geometric View 9
3.2 Finding the Eigendecomposition 10
3.3 Using the Eigendecomposition 10
3.3.1 Example 10
3.4 Principal Components Regression 14

4 Ridge Regression **15**
4.1 Some Words of Advice about Ridge Regression 18
4.2 Penalties vs. Constraints 19
4.3 Ridge Regression in R 19
4.4 Other Penalties/Constraints 20

5 High-Dimensional Regression **20**
5.1 Demo 21

6 Further Reading **23**

1 Why Collinearity Is a Problem

Remember our formula for the estimated coefficients in a multiple linear regression:

$$\widehat{\beta} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y}$$

This is obviously going to lead to problems if $\mathbf{x}^T\mathbf{x}$ isn't invertible. Similarly, the variance of the estimates,

$$\mathrm{Var}\left[\widehat{\beta}\right] = \sigma^2(\mathbf{x}^T\mathbf{x})^{-1}$$

will blow up when $\mathbf{x}^T\mathbf{x}$ is singular. If that matrix isn't exactly singular, but is close to being non-invertible, the variances will become huge.

There are several equivalent conditions for any square matrix, say $\mathbf{u}$, to be singular or non-invertible:

- The determinant $\det \mathbf{u}$ or $|\mathbf{u}|$ is 0.
- At least one eigenvalue[1] of u is 0. (This is because the determinant of a matrix is the product of its eigenvalues.)
- $\mathbf{u}$ is **rank deficient**, meaning that one or more of its columns (or rows) is equal to a linear combination of the other rows[2].

Since we're not concerned with any old square matrix, but specifically with $\mathbf{x}^T\mathbf{x}$, we have an additional equivalent condition:

- $\mathbf{x}$ is **column-rank** deficient, meaning one or more of its columns is equal to a linear combination of the others.

The last explains why we call this problem **collinearity**: it looks like we have p different predictor variables, but really some of them are linear combinations of the others, so they don't add any information. The real number of distinct variables is $q < p$, the column rank of $\mathbf{x}$. If the exact linear relationship holds among more than two variables, we talk about **multicollinearity**; **collinearity** can refer either to the general situation of a linear dependence among the predictors, or, by contrast to multicollinearity, a linear relationship among just two of the predictors.

Again, if there isn't an *exact* linear relationship among the predictors, but they're close to one, $\mathbf{x}^T\mathbf{x}$ will be invertible, but $(\mathbf{x}^T\mathbf{x})^{-1}$ will be huge, and the variances of the estimated coefficients will be enormous. This can make it very hard to say anything at all precise about the coefficients, but that's not *necessarily* a problem.

1.1 Dealing with Collinearity by Deleting Variables

Since not all of the p variables are actually contributing information, a natural way of dealing with collinearity is to drop some variables from the model. If you want to do this, you should think very carefully about *which* variable to delete. As a concrete example: if we try to include all of a student's grades as

[1] You learned about eigenvalues and eigenvectors in linear algebra; if you are rusty, now is an excellent time to refresh your memory.

[2] The equivalence of this condition to the others is not at all obvious, but, again, is proved in linear algebra.

predictors, as well as their over-all GPA, we'll have a problem with collinearity (since GPA is a linear function of the grades). But depending on what we want to predict, it might make more sense to use just the GPA, dropping all the individual grades, or to include the individual grades and drop the average[3].

1.2 Diagnosing Collinearity Among Pairs of Variables

Linear relationships between pairs of variables are fairly easy to diagnose: we make the pairs plot of all the variables, and we see if any of them fall on a straight line, or close to one. Unless the number of variables is huge, this is by far the best method. If the number of variables *is* huge, look at the correlation matrix, and worry about any entry off the diagonal which is (nearly) ± 1.

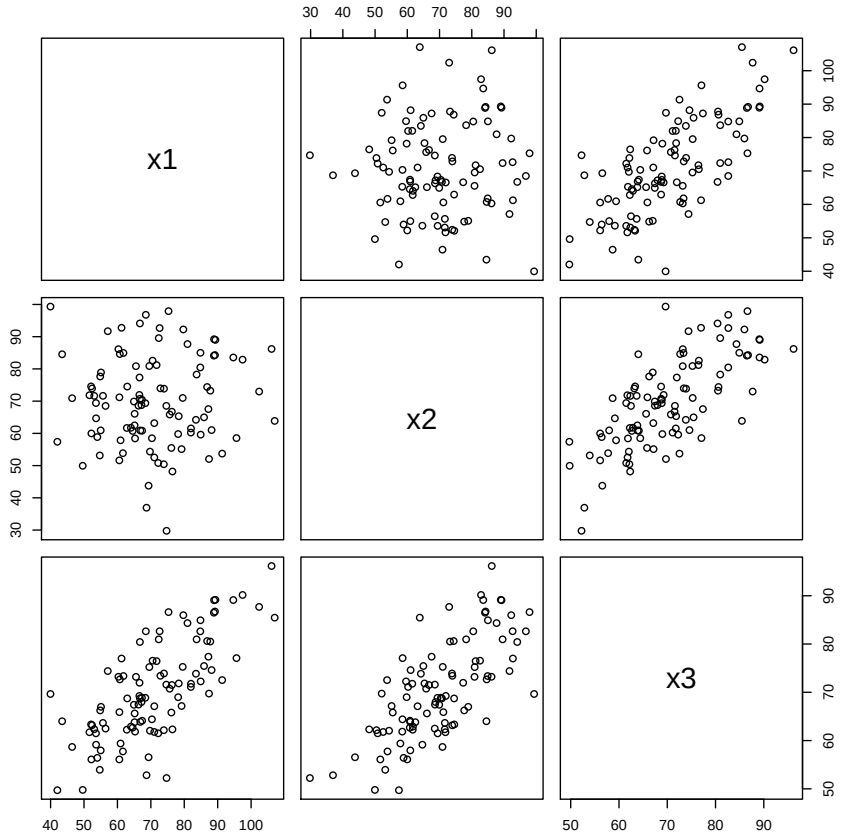
1.3 Why Multicollinearity Is Harder

A multicollinear relationship involving three or more variables might be totally invisible on a pairs plot. For instance, suppose X_1 and X_2 are independent Gaussians, of equal variance σ^2, and X_3 is their average, $X_3 = (X_1 + X_2)/2$. The correlation between X_1 and X_3 is

$$\begin{aligned}
\mathrm{Cor}(X_1, X_3) &= \frac{\mathrm{Cov}\left[X_1, X_3\right]}{\sqrt{\mathrm{Var}\left[X_1\right]\mathrm{Var}\left[X_3\right]}} && (1)\\
&= \frac{\mathrm{Cov}\left[X_1, (X_1+X_2)/2\right]}{\sqrt{\sigma^2\sigma^2/2}} && (2)\\
&= \frac{\sigma^2/2}{\sigma^2/\sqrt{2}} && (3)\\
&= \frac{1}{\sqrt{2}} && (4)
\end{aligned}$$

This is also the correlation between X_2 and X_3. A correlation of $1/\sqrt{2}$ isn't trivial, but is hardly perfect, and doesn't really distinguish itself on a pairs plot (Figure 1).

[3] One could also drop just one of the individual class grades from the average, but it's harder to think of a scenario where that makes sense.




```
##            x1         x2        x3
## x1 1.00000000 0.03788452 0.7250514
## x2 0.03788452 1.00000000 0.7156686
## x3 0.72505136 0.71566863 1.0000000
```

```
# Simulation: two independent Gaussians
x1 <- rnorm(100, mean=70, sd=15)
x2 <- rnorm(100, mean=70, sd=15)
# Add in a linear combination of X1 and X2
x3 <- (x1+x2)/2
pairs(cbind(x1,x2,x3))
cor(cbind(x1,x2,x3))
```

FIGURE 1: *Illustration of a perfect* multi-*collinear relationship might not show up on a pairs plot or in a correlation matrix.*

1.4 Geometric Perspective

The predictors $X_1, \ldots X_p$ form a p-dimensional random vector $\mathbf{X}$. Ordinarily, we expect this random vector to be scattered throughout p-dimensional space. When we have collinearity (or multicollinearity), the vectors are actually confined to a lower-dimensional subspace. The **column rank** of a matrix is the number of linearly independent columns it has. If $\mathbf{x}$ has column rank $q < p$, then the data vectors are confined to a q-dimensional subspace. It *looks* like we've got p different variables, but really by a change of coordinates we could get away with just q of them.

2 Variance Inflation Factors

If the predictors are correlated with each other, the standard errors of the coefficient estimates will be bigger than if the predictors were uncorrelated.

If the predictors were uncorrelated, the variance of $\hat{\beta}_i$ would be

$$\mathrm{Var}\left[\hat{\beta}_i\right] = \frac{\sigma^2}{n s^2_{X_i}} \tag{5}$$

just as it is in a simple linear regression. With correlated predictors, however, we have to use our general formula for the least squares:

$$\mathrm{Var}\left[\hat{\beta}_i\right] = \sigma^2 (\mathbf{x}^T\mathbf{x})^{-1}_{i+1,i+1} \tag{6}$$

(Why are the subscripts on the matrix $i+1$ instead of i?) The ratio between Eqs. 6 and 5 is the **variance inflation factor** for the i^{th} coefficient, VIF_i. The average of the variance inflation factors across all predictors is often written $\overline{VIF}$, or just VIF.

Folklore says that $VIF_i > 10$ indicates "serious" multicollinearity for the predictor. I have been unable to discover who first proposed this threshold, or what the justification for it is. It is also quite unclear what to do about this. Large variance inflation factors do not, after all, violate any model assumptions.

2.1 Why $VIF_i \geq 1$

Let's take the case where $p = 2$, so $\mathbf{x}^T\mathbf{x}$ is a 3×3 matrix. As you saw in the homework,

$$\frac{1}{n}\mathbf{x}^T\mathbf{x} = \begin{bmatrix} 1 & \overline{x_1} & \overline{x_2} \\ \overline{x_1} & \overline{x_1^2} & \overline{x_1 x_2} \\ \overline{x_2} & \overline{x_1 x_2} & \overline{x_2^2} \end{bmatrix}$$

After tedious but straightforward algebra[4], we get for the inverse (deep breath)

$$\left(\frac{1}{n}\mathbf{x}^T\mathbf{x}\right)^{-1} = \frac{1}{\widehat{\mathrm{Var}}[X_1]\widehat{\mathrm{Var}}[X_2] - \widehat{\mathrm{Cov}}[X_1,X_2]^2} \begin{bmatrix} \widehat{\mathrm{Var}}[X_1]\widehat{\mathrm{Var}}[X_2] - \widehat{\mathrm{Cov}}[X_1,X_2]^2 + \widehat{\mathrm{Var}}[\overline{x_2}X_1 - \overline{x_1}X_2] & \overline{x} \\ \overline{x_1}\widehat{\mathrm{Var}}[X_2] - \widehat{\mathrm{Cov}}[X_1,X_2]\,\overline{x_2} & \\ \overline{x_1}\widehat{\mathrm{Cov}}[X_1,X_2] - \widehat{\mathrm{Var}}[X_1]\,\overline{x_2} & \end{bmatrix}$$

where the hats on the variances and covariances indicate that they are sample, not population, quantities.

Notice that the pre-factor to the matrix, which is the determinant of $n^{-1}\mathbf{x}^T\mathbf{x}$, blows up when X_1 and X_2 are either perfectly correlated or perfectly anti-correlated — which is as it should be, since then we'll have exact collinearity.

The variances of the estimated slopes are, using this inverse,

$$\mathrm{Var}\left[\hat{\beta}_1\right] = \frac{\sigma^2}{n}\frac{\widehat{\mathrm{Var}}[X_2]}{\widehat{\mathrm{Var}}[X_1]\widehat{\mathrm{Var}}[X_2] - \widehat{\mathrm{Cov}}[X_1,X_2]^2} = \frac{\sigma^2}{n(\widehat{\mathrm{Var}}[X_1] - \widehat{\mathrm{Cov}}[X_1,X_2]^2/\widehat{\mathrm{Var}}[X_2])}$$

and

$$\mathrm{Var}\left[\hat{\beta}_2\right] = \frac{\sigma^2}{n}\frac{\widehat{\mathrm{Var}}[X_1]}{\widehat{\mathrm{Var}}[X_1]\widehat{\mathrm{Var}}[X_2] - \widehat{\mathrm{Cov}}[X_1,X_2]^2} = \frac{\sigma^2}{n(\widehat{\mathrm{Var}}[X_2] - \widehat{\mathrm{Cov}}[X_1,X_2]^2/\widehat{\mathrm{Var}}[X_1])}$$

Notice that if $\widehat{\mathrm{Cov}}[X_1,X_2] = 0$, these reduce to

$$\mathrm{Var}\left[\hat{\beta}_1\right] = \frac{\sigma^2}{n\widehat{\mathrm{Var}}[X_1]}\ ,\ \mathrm{Var}\left[\hat{\beta}_2\right] = \frac{\sigma^2}{n\widehat{\mathrm{Var}}[X_2]}$$

exactly as we'd see in simple linear regressions. When covariance is present, however, regardless of its sign, it increases the variance of the estimates.

With a great deal of even more tedious algebra, it can be shown that this isn't just a weird fact about the $p = 2$ case, but is true generically. The variance inflation factor for X_i can be found by regressing X_i on all of the other X_j, computing the R^2 of this regression[5], say R_i^2, and setting $VIF_i = 1/(1 - R_i^2)$.[6] The consequence is that $VIF_i \geq 1$, with the variance inflation factor increasing as X_i becomes more correlated with some linear combination of the other predictors.

[4] At least, if you remember how to calculate the determinant of a matrix, a matter on which I evidently had a brain-fault this afternoon.

[5] I'd admit this was an exception to my claim that R^2 is at best useless, except that we can get the exact same number, without running all these regressions, just by inverting $\mathbf{x}^T\mathbf{x}$.

[6] The trick to showing this involves relating the co-factors which appear when we're inverting $n^{-1}\mathbf{x}^T\mathbf{x}$ to the coefficients in the regression of X_i on all the other X_j, followed by a mess of book-keeping.

3 Matrix-Geometric Perspective on Multicollinearity

Multicollinearity means that there exists (at least) one set of constants $a_0, a_1, \ldots a_p$, $a_1, \ldots a_p$ not all zero, such that

$$a_1 X_1 + a_2 X_2 + \ldots a_p X_p = \sum_{i=1}^{p} a_i X_i = a_0$$

To simplify this, let's introduce the $p \times 1$ matrix $\mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_p \end{bmatrix}$, so we can write multicollinearity as

$$\mathbf{a}^T \mathbf{X} = a_0$$

for $\mathbf{a} \neq \mathbf{0}$.

If this equation holds, then

$$\mathrm{Var}\left[\mathbf{a}^T \mathbf{X}\right] = \mathrm{Var}\left[\sum_{i=1}^{p} a_i X_i\right] = \mathrm{Var}\left[a_0\right] = 0$$

Conversely, if $\mathrm{Var}\left[\mathbf{a}^T \mathbf{X}\right] = 0$, then $\mathbf{a}^T \mathbf{X}$ must be equal to some constant, which we can call a_0. So multicollinearity is equivalent to the existence of a vector $\mathbf{a} \neq \mathbf{0}$ where

$$\mathrm{Var}\left[\mathbf{a}^T \mathbf{X}\right] = 0$$

I make these observations because we are old hands now at the variances of weighted sums.

$$\begin{aligned} \mathrm{Var}\left[\mathbf{a}^T \mathbf{X}\right] &= \mathrm{Var}\left[\sum_{i=1}^{p} a_i X_i\right] && (7) \\ &= \sum_{i=1}^{p} \sum_{j=1}^{p} a_i a_j \mathrm{Cov}\left[X_i, X_j\right] && (8) \\ &= \mathbf{a}^T \mathrm{Var}\left[\mathbf{X}\right] \mathbf{a} && (9) \end{aligned}$$

Multicollinearity therefore means the equation

$$\mathbf{a}^T \mathrm{Var}\left[\mathbf{X}\right] \mathbf{a} = 0$$

has a solution $\mathbf{a} \neq 0$.

Solving a quadratic equation in matrices probably does not sound like much fun, but this is where we appeal to results in linear algebra[7]. $\mathrm{Var}\left[\mathbf{X}\right]$ is a very special matrix: it is square ($p \times p$), symmetric, and positive-definite, meaning

[7] This is also a big part of why we make you take linear algebra.

that $\mathbf{a}^T \mathrm{Var}[\mathbf{X}]\,\mathbf{a} \geq 0$. (Since, after all, that expression is the variance of the scalar $\sum_{i=1}^{p} a_i X_i$, and variances of scalars are ≥ 0.) We may therefore appeal to the *spectral* or *eigendecomposition* theorem of linear algebra to assert the following:

1. There are p different $p \times 1$ vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_p$, the **eigenvectors** of $\mathrm{Var}[\mathbf{X}]$, such that
$$\mathrm{Var}[\mathbf{X}]\,\mathbf{v}_i = \lambda_i \mathbf{v}_i$$
for scalar constants $\lambda_1, \lambda_2, \ldots \lambda_p$, the **eigenvalues** of $\mathrm{Var}[\mathbf{X}]$. The ordering of the eigenvalues and eigenvectors is arbitrary, but it is conventional to arrange them so that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$.

2. The eigenvalues are all ≥ 0. (Some of them may be equal to each other; these are called **repeated**, **multiple** or **degenerate** eigenvalues.)

3. The eigenvectors can be chosen so that they all have length 1, and are orthogonal to each other, so $\mathbf{v}_j^T \mathbf{v}_i = \delta_{ij}$.

4. Any vector can be re-written as a sum of eigenvectors:
$$\mathbf{a} = \sum_{i=1}^{p} (\mathbf{a}^T \mathbf{v}_i)\mathbf{v}_i$$
(Here I have used the parentheses to eliminate any ambiguity about the order in which the matrices are to be multiplied; $\mathbf{a}^T \mathbf{v}_i$ is always a scalar.)

5. $\mathrm{Var}[\mathbf{X}]$ can be expressed as
$$\mathrm{Var}[\mathbf{X}] = \mathbf{V}\mathbf{D}\mathbf{V}^T$$
where $\mathbf{V}$ is the matrix whose i^{th} column is $\mathbf{v}_i$, (and so $\mathbf{V}^T$ is the matrix where $\mathbf{v}_i$ is the i^{th} column), and $\mathbf{D}$ is the diagonal matrix whose entries are $\lambda_1, \lambda_2, \ldots \lambda_p$.

Suppose that one or more of the eigenvalues are zero. Since we've put them in order, this means that the positive eigenvalues are $\lambda_1, \ldots \lambda_q$ (for some $q < p$), and $\lambda_{q+1}, \ldots \lambda_p$ are all zero. It follows that $\mathbf{v}_{q+1}, \ldots \mathbf{v}_p$ all give us linear combinations of the X_i which are multicollinear. So a sufficient condition for multicollinearity is that $\mathrm{Var}[\mathbf{X}]$ have zero eigenvalues.

Conversely, suppose $\mathbf{a}^T \mathrm{Var}[\mathbf{X}]\,\mathbf{a} = 0$, and $\mathbf{a} \neq \mathbf{0}$. Let's re-express this using

the eigendecomposition.

$$\begin{aligned}
\mathrm{Var}[\mathbf{X}]\,\mathbf{a} &= \mathrm{Var}[\mathbf{X}]\sum_{i=1}^{p}(\mathbf{a}^T\mathbf{v}_i)\mathbf{v}_i && (10)\\
&= \sum_{i=1}^{p}(\mathbf{a}^T\mathbf{v}_i)\mathrm{Var}[\mathbf{X}]\,\mathbf{v}_i && (11)\\
&= \sum_{i=1}^{p}(\mathbf{a}^T\mathbf{v}_i)\lambda_i\mathbf{v}_i && (12)\\
\mathbf{a}^T\mathrm{Var}[\mathbf{X}]\,\mathbf{a} &= \left(\sum_{j=1}^{p}(\mathbf{a}^T\mathbf{v}_j)\mathbf{v}_j\right)^T\sum_{i=1}^{p}(\mathbf{a}^T\mathbf{v}_i)\lambda_i\mathbf{v}_i && (13)\\
&= \sum_{i=1}^{p}\sum_{j=1}^{p}(\mathbf{a}^T\mathbf{v}_i)(\mathbf{a}^T\mathbf{v}_j)\mathbf{v}_j^T\mathbf{v}_i && (14)\\
&= \sum_{i=1}^{p}(\mathbf{a}^T\mathbf{v}_i)^2\lambda_i && (15)
\end{aligned}$$

Since $(\mathbf{a}^T\mathbf{v}_i)^2 \geq 0$, the only way the whole sum can be zero is if $(\mathbf{a}^T\mathbf{v}_i)^2 > 0$ only when $\lambda_i = 0$.

We have therefore established the following:

1. The predictors are multi-collinear if and only if $\mathrm{Var}[\mathbf{X}]$ has zero eigenvalues.
2. Every multi-collinear combination of the predictors is either an eigenvector of $\mathrm{Var}[\mathbf{X}]$ with zero eigenvalue, or a linear combination of such eigenvectors.

3.1 The Geometric View

Every eigenvector of $\mathrm{Var}[\mathbf{X}]$ points out a direction in the space of predictors. The leading eigenvector $\mathbf{v}_1$, the one going along with the largest eigenvalue, points out the direction of highest variance (and that variance is λ_1). The next-to-leading eigenvector, $\mathbf{v}_2$, points out the direction orthogonal to $\mathbf{v}_1$ which has the highest variance, and so forth down the line. The eigenvectors of $\mathrm{Var}[\mathbf{X}]$ are also called the **principal components** of the predictors, because of their role as the directions of maximum variance.

The eigenvectors going along with zero eigenvalues point out directions in the predictor space along which there is no variance, precisely because those directions amount to weighted sums of the original variables which equal constants. The q non-zero eigenvalues mark out the q-dimensional subspace in which all the data vectors lie. If $q < p$, then the predictors are rank-deficient, and the rank of $\mathbf{x}$ is just q.

3.2 Finding the Eigendecomposition

Because finding eigenvalues and eigenvectors of matrices is so useful for so many situations, mathematicians and computer scientists have devoted incredible efforts over the last two hundred years to fact, precise algorithms for computing them. This is not the place to go over how those algorithms work; it is the place to say that much of the fruit of those centuries of effort is embodied in the linear algebra packages R uses. Thus, when you call

```
eigen(A)
```

you get back a list, containing the eigenvalues of the matrix `A` (in a vector), and its eigenvectors (in a matrix), and this is both a very fast and a very reliable calculation. If your matrix has very special structure (e.g., it's sparse, meaning almost all its entries are zero), there are more specialized packages adapted to your needs, but we don't pursue this further here; for most data-analytic purposes, ordinary `eigen` will do.

3.3 Using the Eigendecomposition

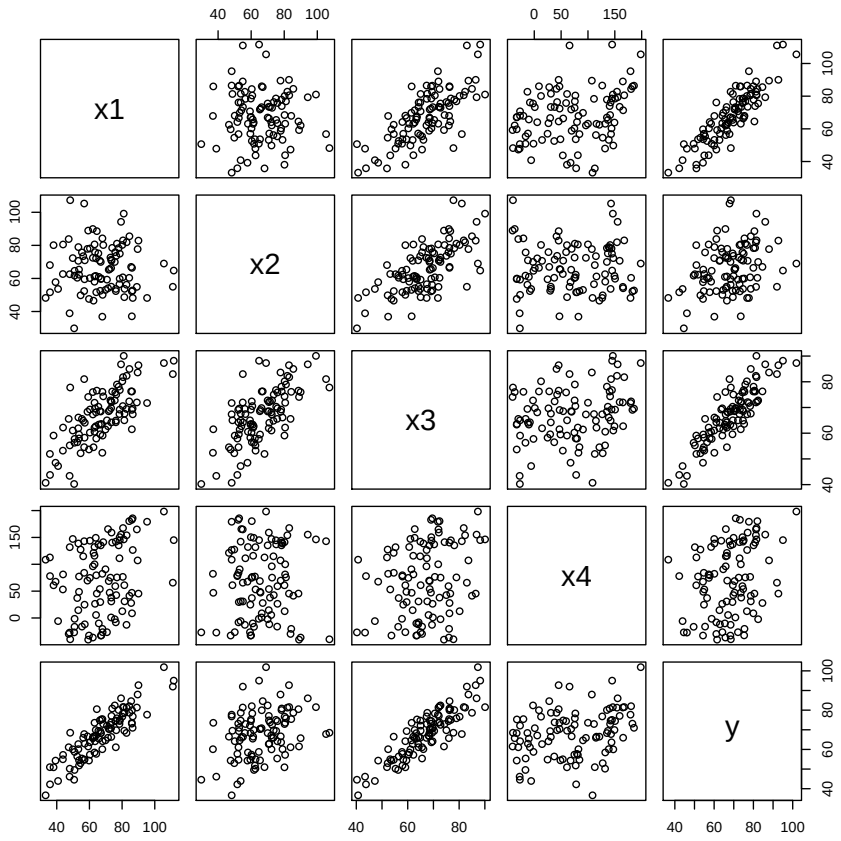
1. Find the eigenvalues and eigenvectors.
2. If any eigenvalues are zero, the data is multicollinear; if any are very close to zero, the data is nearly multicollinear.
3. Examine the corresponding eigenvectors. These indicate the linear combinations of variables which equal constants (or are nearly constant if the eigenvalue is only nearly zero). Ideally, these will be combinations of a reasonably small number of variables (i.e., most of the entries in the eigenvector will be zero), so you can ask whether there are substantive reasons to delete one or more of those predictors.

3.3.1 Example

I'll make up some data which displays exact multi-collinearity. Let's say that X_1 and X_2 are both Gaussian with mean 70 and standard deviation 15, and are uncorrelated; that $X_3 = (X_1 + X_2)/2$; and that $Y = 0.7X_1 + 0.3X_2 + \epsilon$, with $\epsilon \sim N(0, 15)$.

```
# Simulation: two independent Gaussians
x1 <- rnorm(100, mean=70, sd=15)
x2 <- rnorm(100, mean=70, sd=15)
# Add in a linear combination of X1 and X2
x3 <- (x1+x2)/2
# X4 is somewhat correlated with X1 but not relevant to Y
x4 <- x1+runif(100,min=-100,max=100)
# Y is a linear combination of the X's plus noise
y <- 0.7*x1 + 0.3*x2 + rnorm(100, mean=0, sd=sqrt(15))
df <- data.frame(x1=x1, x2=x2, x3=x3, x4=x4, y=y)
```

FIGURE 2: *Small simulation illustrating exact collinearity.*




```
##             x1          x2        x3         x4        y
## x1  1.00000000 -0.01979669 0.7290418 0.29354541 0.8810356
## x2 -0.01979669  1.00000000 0.6699024 0.03450894 0.3263256
## x3  0.72904178  0.66990244 1.0000000 0.24161019 0.8776559
## x4  0.29354541  0.03450894 0.2416102 1.00000000 0.3006694
## y   0.88103556  0.32632563 0.8776559 0.30066941 1.0000000
```

```
pairs(df)
cor(df)
```

FIGURE 3: *Pairs plot and correlation matrix for the example of Figure 2. Notice that neither the pairs plot nor the correlation matrix reveals a problem, which is because it only arises when considering X_1, X_2, X_3 at once.*

```
# Create the variance matrix of the predictor variables
var.x <- var(df[,c("x1","x2","x3","x4")])
# Find the eigenvalues and eigenvectors
var.x.eigen <- eigen(var.x)
# Which eigenvalues are (nearly) 0?
(zero.eigenvals <- which(var.x.eigen$values < 1e-12))
```

```
## [1] 4
```

```
# Display the corresponding vectors
(zero.eigenvectors <- var.x.eigen$vectors[,zero.eigenvals])
```

```
## [1]  4.082483e-01  4.082483e-01 -8.164966e-01  3.330669e-16
```

FIGURE 4: *Example of using the eigenvectors of* $\mathrm{Var}[X]$ *to find collinear combinations of the predictor variables. Here, what this suggests is that* $-X_1 - X_2 + 2X_3 = \text{constant}$. *This is correct, since* $X_3 = (X_1 + X_2)/2$, *but the eigen-decomposition didn't* know *this; it discovered it.*

3.4 Principal Components Regression

Let's define some new variables:

$$W_1 = \mathbf{v}_1^T X \quad (16)$$
$$W_i = \mathbf{v}_i^T X \quad (17)$$
$$W_p = \mathbf{v}_p^T X \quad (18)$$
$$(19)$$

W_1 is the projection of the original data vector X onto the leading eigenvector, or the **principal component**. It is called the **score** on the first principal component. It has a higher (sample) variance than any other linear function of the original predictors. W_2 is the projection or score on the second principle component. It has more variance than any other linear combination of the original predictors which is uncorrelated with W_1. In fact, $\widehat{\mathrm{Cov}}[W_i, W_j] = 0$ if $i \neq j$.

In **principle components regression**, we pick some $k \leq p$ and use the model

$$Y = \gamma_0 + \gamma_1 W_1 + \ldots \gamma_k W_k + \epsilon$$

where as usual we presume ϵ has expectation 0, constant variance, and no correlation from one observation to another. (I use the Greek letter γ, instead of β, to emphasize that these coefficients are *not* directly comparable to those of our original linear model.) We are regressing not on our original variables, but on uncorrelated linear combinations of those variables.

If $k = p$, then we get exactly the same predictions and fitted values as in the original linear model, though the coefficients aren't the same. This would amount to doing a change of coordinates in the space of predictors, so that all of the new coordinates were uncorrelated, but wouldn't otherwise change anything.

If there are only $q < p$ non-zero eigenvalues, we should not use $k > q$. Using $k = q$ uses all the linear combinations of the original predictors which aren't collinear. However, we might deliberately pick $k < q$ so as to simplify our model. As I said above, the principal components are the directions in the predictor space with the highest variance, so by using a small k we confine ourselves to those directions, and ignore all the other aspects of our original predictors. This may introduce bias, but should reduce the variance in our predictions. (Remember that the variance in our coefficient estimates, and so in our predictions, goes down with the variance of the predictor variables.)

There are a number of things to be said about principal components regression.

1. We need some way to pick k. The in-sample MSE will decrease as k grows (why?), but this might not be a good guide to out-of-sample predictions, or to whether the modeling assumptions are fulfilled.

2. The PC regression can be hard to interpret.

The last point needs some elaboration. Each one of the principal components is a linear combination of the original variables. Sometimes these are easy to interpret, other times their meaning (if they have any) is thoroughly obscure. Whether this matters depends very much on how deeply committed you are to interpreting the coefficients.

As for picking k, there are two (potentially) rival objectives. One is to pick the number of components which will let us predict well. The in-sample mean squared error *has* to decrease as k grows, so we would really like some measure of actual out-of-sample or generalization error; the cross-validation method I will describe below is applicable, but there are other potentially-applicable techniques. The other objective is to have a set of variables which satisfy the assumptions of the multiple linear regression model. In my experience, it is not very common for principal components regression to actually satisfy the modeling assumptions, but it can work surprisingly well as a predictive tool anyway.

4 Ridge Regression

The real problem with collinearity is that when it happens, there isn't a *unique* solution to the estimating equations. There are rather infinitely many solutions, which all give the minimum mean squared error. It feels perverse, somehow, to get rid of predictors because they give us too many models which fit too well. A better response is to pick *one* of these solutions, by adding some other criterion we'd like to see in the model.

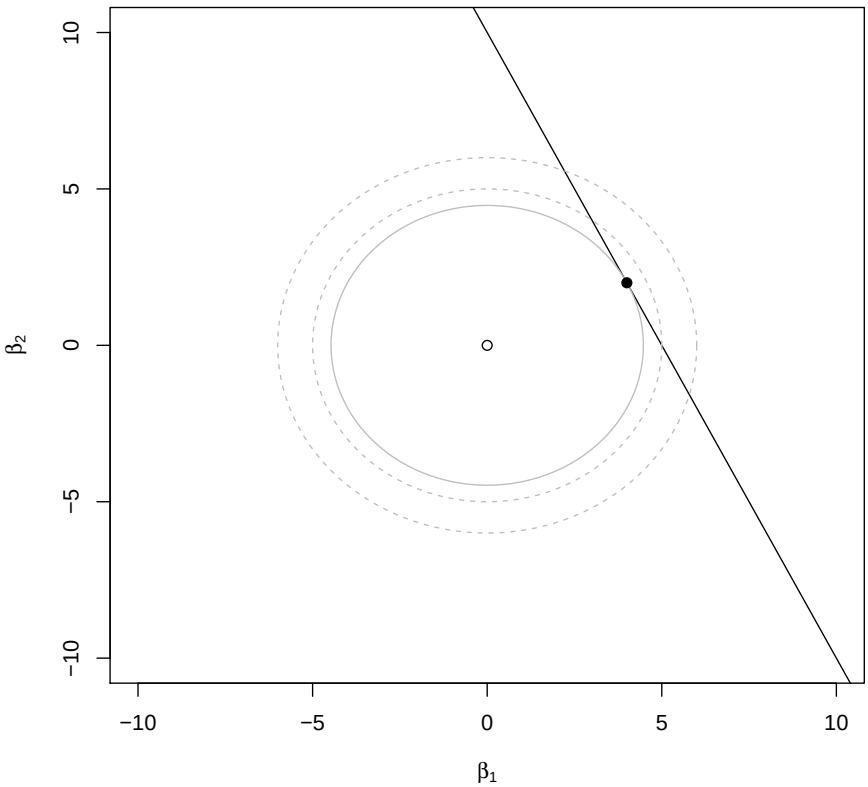
There are many ways to do this, but one which works well in practice is the following: all else being equal, we prefer models with smaller slopes, ones closer to zero. Specifically, let's say that we prefer the length of the coefficient vector, $\|\beta\|$, to be small. Now, at least abstractly, we have a situation like that shown in Figure 5. The black line marks out all of the β_1, β_2 combinations which give us exactly the same mean squared error. They all give the same error because of a collinearity between X_1 and X_2. But there is a single point on the black line which comes closest to the origin — it touches the solid grey circle. Other points on the line, while they have equal MSEs, have larger $\|\beta\|$ (they lie on one of the dashed grey circles), so we don't use them.

What if everything else isn't equal? (Say, for instance, that the data are only *nearly* collinear.) We'll need some way to trade off having a smaller MSE against having a smaller vector of coefficients. Since we're looking at *squared* error, I hope it is somewhat plausible that we should also look at the *squared* length of the coefficient vector; if you don't buy that, you can at least take my word for it that it simplifies the math.

Specifically, let's replace our old optimization problem

$$\min \frac{1}{n}(\mathbf{y} - \mathbf{x}\mathbf{b})^T(\mathbf{y} - \mathbf{x}\mathbf{b})$$

```
# Add a circle to an existing plot
  # R, bizarrely, does not have any built-in function for this
# Inputs: x coordinate of center; y coordinate of center; radius;
  # number of equiangular steps; additional graphical parameters
# Outputs: none
# Side-effects: a circle is added to the existing plot
circle <- function(x0, y0, r, n=1000, ...) {
    theta <- seq(from=0, to=2*pi, length.out=n)  # Angles
    x <- x0 + r*cos(theta)                       # x coordinates
    y <- y0 + r*sin(theta)                       # y coordinates
    lines(x,y,...) # Draw the lines connecting all the points, in order
}
plot(0,type="n",xlab=expression(beta[1]),ylab=expression(beta[2]),
     xlim=c(-10,10), ylim=c(-10,10))
abline(a=10,b=-2)
points(0,0)
circle(0,0,sqrt(20),col="grey")
points(4,2,col="black",pch=19)
circle(0,0,5,col="grey",lty="dashed")
circle(0,0,6,col="grey",lty="dashed")
```



00:28 Thursday 29th October, 2015

FIGURE 5: *Geometry of ridge regression when the predictors are collinear. The black line shows all the combinations of β_1 and β_2 which minimize the MSE. We chose the coefficient vector (the black point) which comes closest to the origin (the dot). Equivalently, this is the parameter vector with the smallest MSE among all the points at equal distance from the origin (solid grey circle). Other coefficient vectors either have a worse MSE (they don't lie on the black line), or are further from the origin (they lie on one of the dashed grey circles).*

with a new, *penalized* optimization problem

$$\min \frac{1}{n}(\mathbf{y}-\mathbf{x}\mathbf{b})^T(\mathbf{y}-\mathbf{x}\mathbf{b}) - \frac{\lambda}{n}\|\mathbf{b}\|^2$$

Here the **penalty factor** $\lambda > 0$ tells us the rate at which we're willing to make a trade-off between having a small mean squared error and having a short vector of coefficients[8]. We'd accept $\|\mathbf{b}\|^2$ growing by 1 if it reduced the MSE by more than λ/n. We'll come back later to how to pick λ.

It's easy to pose this optimization problem: can we actually solve it, and is the solution good for anything? Solving is actually straightforward. We can re-write $\|\mathbf{b}\|^2$ as, in matrix notation, $\mathbf{b}^T\mathbf{b}$, so the gradient is

$$\nabla_{\mathbf{b}}\left(\frac{1}{n}(\mathbf{y}-\mathbf{x}\mathbf{b})^T(\mathbf{y}-\mathbf{x}\mathbf{b}) + \frac{\lambda}{n}\mathbf{b}^T\mathbf{b}\right) = \frac{2}{n}\left(-\mathbf{x}^T\mathbf{y} + \mathbf{x}^T\mathbf{x}\mathbf{b} + \lambda\mathbf{b}\right)$$

Set this to zero at the optimum, $\tilde{\beta}_\lambda$,

$$\mathbf{x}^T\mathbf{y} = (\mathbf{x}^T\mathbf{x} + \lambda\mathbf{I})\tilde{\beta}_\lambda$$

and solve:

$$\tilde{\beta}_\lambda = (\mathbf{x}^T\mathbf{x} + \lambda\mathbf{I})^{-1}\mathbf{x}^T\mathbf{y}$$

Notice what we've done here: we've taken the old matrix $\mathbf{x}^T\mathbf{x}$ and we've added λ to every one of its diagonal entries. (This is the "ridge" that gives ridge regression its name.) If the predictor variables were centered, this would amount to estimating the coefficients as though each of them as had a little bit more variance than they really did, while leaving all the covariances alone. This would break any exact multicollinearity, so the inverse always exists, and there is always some solution.

What about the intercept? The intercept is different from the other coefficients; it's just a fudge factor we put in to make sure that the regression line goes through the mean of the data. It doesn't make as much sense to penalize its length, so ridge regression is usually done after centering all the variables, both the predictors and the response. This doesn't change the slopes, but sets the intercept to zero. Then, after we have $\tilde{\beta}_\lambda$, we get the intercept by plugging it in to $\overline{y} = \overline{x}\tilde{\beta}_\lambda$.

There are two prices to doing this.

1. We need to pick λ (equivalently, c) somehow.
2. Our estimates of the coefficients are no longer unbiased, but are "shrunk" towards zero.

[8] $\lambda = 0$ means we ignore the length of the coefficient vector and we're back to ordinary least squares. $\lambda < 0$ would mean we *prefer* larger coefficients, and would lead to some truly perverse consequences.

Point (2) is not as bad as it might appear. If λ is fixed, and we believe our modeling assumptions, we can calculate the bias and variance of the ridge estimates:

$$\mathbb{E}\left[\tilde{\beta}_\lambda\right] = \left(\mathbf{x}^T\mathbf{x} + \lambda\mathbf{I}\right)^{-1}\mathbf{x}^T\mathbb{E}\left[\mathbf{Y}\right] \quad (20)$$

$$= \left(\mathbf{x}^T\mathbf{x} + \lambda\mathbf{I}\right)^{-1}\mathbf{x}^T\mathbf{x}\beta \quad (21)$$

$$\mathrm{Var}\left[\tilde{\beta}_\lambda\right] = \mathrm{Var}\left[\left(\mathbf{x}^T\mathbf{x} + \lambda\mathbf{I}\right)^{-1}\mathbf{x}^T\mathbf{Y}\right] \quad (22)$$

$$= \mathrm{Var}\left[\left(\mathbf{x}^T\mathbf{x} + \lambda\mathbf{I}\right)^{-1}\mathbf{x}^T\epsilon\right] \quad (23)$$

$$= \left(\mathbf{x}^T\mathbf{x} + \lambda\mathbf{I}\right)^{-1}\mathbf{x}^T\sigma^2\mathbf{I}\mathbf{x}\left(\mathbf{x}^T\mathbf{x} + \lambda\mathbf{I}\right)^{-1} \quad (24)$$

$$= \sigma^2\left(\mathbf{x}^T\mathbf{x} + \lambda\mathbf{I}\right)^{-1}\mathbf{x}^T\mathbf{x}\left(\mathbf{x}^T\mathbf{x} + \lambda\mathbf{I}\right)^{-1} \quad (25)$$

Notice how both of these expressions smoothly approach the corresponding formulas ones for ordinary least squares as $\lambda \to 0$. Indeed, under the Gaussian noise assumption, $\tilde{\beta}_\lambda$ actually has a Gaussian distribution with the given expectation and variance.

Of course, if λ is not fixed in advance, we'd have to worry about the randomness in the distribution of λ. A common practice here is **data splitting**: randomly divide the data into two parts, and use one to pick λ and the other to then actually estimate the parameters, which will have the stated bias and standard errors. (Typically, but not necessarily, the two parts of the data are equally big.)

As for point (1), picking λ, this is also a solvable problem. The usual approach is **cross-validation**: trying a lot of different values of λ, estimate the model on all but one data point, and then see how well different λ's predict that held-out data point. Since there's nothing special about one data point rather than another, do this for each data point, and average the out-of-sample squared errors. Pick the λ which does best at predicting data it didn't get to see. (There are lots of variants, some of which we'll cover later in the course.)

4.1 Some Words of Advice about Ridge Regression

Units and Standardization If the different predictor variables don't have physically comparable units[9], it's a good idea to standardize them first, so they all have mean 0 and variance 1. Otherwise, penalizing $\beta^T\beta = \sum_{i=1}^{p}\beta_i^2$ seems to be adding up apples, oranges, and the occasional bout of regret. (Some people like to pre-standardize even physically comparable predictors.)

[9] E.g., if they're all masses expressed in grams, they're comparable; if some masses are in kilograms or pounds, they're not comparable but they could easily be made so; if some of them are lengths or prices, they're not physically comparable no matter what units you use.

Stabilization I've presented ridge regression as a way of dealing with multicollinearity, which it is, but it's also perfectly possible to use it when that isn't an issue. The goal there is to **stabilize** the estimates — to reduce their variance, at the cost of a bit of bias. If the linear model is perfectly well-specified, there's little point to doing this, but it can often improve predictions a lot when the model is mis-specified.

4.2 Penalties vs. Constraints

I explained ridge regression above as applying a penalty to long coefficient vectors. There is an alternative perspective which is mathematically equivalent, where instead we *constrain* the length of the coefficient vector.

To see how this works, let's start by setting up the problem: pick some $c > 0$, and then ask for

$$\min_{b \,:\, \|b\| \leq c} \frac{1}{n}(\mathbf{y} - \mathbf{x}\mathbf{b})^T(\mathbf{y} - \mathbf{x}\mathbf{b})$$

Since $\|b\| \leq c$ if and only if $\|b\|^2 \leq c^2$, we might as well say

$$\min_{b \,:\, \|b\|^2 \leq c^2} \frac{1}{n}(\mathbf{y} - \mathbf{x}\mathbf{b})^T(\mathbf{y} - \mathbf{x}\mathbf{b})$$

At this point, we invoke the magic of Lagrange multipliers[10]: we can turn a constrained problem into an unconstrained problem with an additional term, and an additional variable:

$$\min_{\mathbf{b},\lambda} \frac{1}{n}(\mathbf{y} - \mathbf{x}\mathbf{b})^T(\mathbf{y} - \mathbf{x}\mathbf{b}) + \lambda(\mathbf{b}^T\mathbf{b} - c^2)$$

Minimizing over λ means that either $\lambda = 0$, or $\mathbf{b}^T\mathbf{b} = c^2$. The former situation will apply when the unconstrained minimum is within the ball $\|b\| \leq c$; otherwise, the constraint will "bite", and λ will take a non-zero value to enforce it. As c grows, the required constraint λ will become smaller[11].

When we minimize over $\mathbf{b}$, the precise value of c^2 doesn't matter; only λ does. If we know λ, then we are effectively just solving the problem

$$\min_{\mathbf{b}} \frac{1}{n}(\mathbf{y} - \mathbf{x}\mathbf{b})^T(\mathbf{y} - \mathbf{x}\mathbf{b}) + \lambda\mathbf{b}^T\mathbf{b}$$

which is the penalized regression problem we solved before.

4.3 Ridge Regression in R

There are several R implementations of ridge regression; the `MASS` package contains one, `lm.ridge`, which needs you to specify λ. The `ridge` package (Cule, 2014) has `linearRidge`, which gives you the option to set λ, or to select it automatically via cross-validation. (See the next section for a demo of this in action.) There are probably others I'm not aware of.

[10] See further reading, if you have forgotten about Lagrange multipliers.

[11] In economic terms, λ is the internal or "shadow" price we'd pay, in units of MSE, to loosen the constraint.

4.4 Other Penalties/Constraints

Ridge regression penalizes the mean squared error with $\|b\|^2$, the squared length of the coefficient vector. This suggests the idea of using some other measure of how big the vector is, some other **norm**. A mathematically popular family of norms are the ℓ_q norms[12], defined as

$$\|b\|_q = \left(\sum_{i=1}^{p} |b_i|^q\right)^{1/q}$$

The usual Euclidean length is ℓ_2, while ℓ_1 is

$$\|b\|_1 = \sum_{i=1}^{p} |b_i|$$

and (by continuity $\|b\|_0$ is just the number of non-zero entries in b. When $q \neq 2$, penalizing the $\|b\|_q$ does not, usually, have a nice closed-form solution like ridge regression does. Finding the minimum of the mean squared error under an ℓ_1 penalty is called **lasso regression** or the **lasso estimator**, or just **the lasso**[13]. This has the nice property that it tends to give *sparse* solutions — it sets coefficients to be exactly zero (unlike ridge). There are no closed forms for the lasso, but there are efficient numerical algorithms. Penalizing ℓ_0, the number of non-zero coefficients, *sounds* like a good idea, but there are, provably, no algorithms which work substantially faster than trying all possible combinations of variables.

5 High-Dimensional Regression

One situation where we know that we will always have multicollinearity is when $n < p$. After all, n points always define a linear subspace of (at most) $n - 1$ dimensions[14]. When the number of predictors we measure for each data point is bigger than the number of data points, the predictors *have* to be collinear, indeed multicollinear. We are then said to be in a **high-dimensional** regime.

This is an increasingly common situation in data analysis. A very large genetic study might sequence the genes of, say, 500 people — but measure 500,000 genetic markers in each person[15]. If we want to predict some characteristic of the people from the genes (say their height, or blood pressure, or how quickly they would reject a transplanted organ), there is simply no way to estimate

[12] Actually, people usually call them the ℓ_p norms, but we're using p for the number of predictor variables.

[13] Officially, "lasso" here is an acronym for "least angle selection and shrinkage operator". If you believe that phrase came before the acronym, I would like your help in getting some money out of Afghanistan.

[14] Two points define a line, unless the points coincide; three points define a plane, unless the points fall on a line; etc.

[15] I take these numbers, after rounding, from an actual study done in the CMU statistics department a few years ago.

a model by ordinary least squares. Any approach to high-dimensional regression *must* involve either reducing the number of dimensions, until it's $< n$ (as in principle components regression), or penalizing the estimates to make them stable and regular (as in ridge regression), or both.

There is a bit of a myth in recent years that "big data" will solve all our problems, by letting us make automatic predictions about everything without any need for deep understanding. The truth is almost precisely the opposite: when we can measure everything about everyone, p/n blows up, and we are in desperate need of ways of filtering the data and/or penalizing our models. Blindly relying on generic methods of dimension reduction or penalization is going to impose all sorts of bizarre biases, and will work much worse than *intelligent* dimension reduction and *appropriate* penalties, based on actual understanding.

5.1 Demo

Let's apply ridge regression to the simulated data already created, where one predictor variable (X_3) is just the average of two others (X_1 and X_2).

```
library(ridge)
# Fit a ridge regression
  # lambda="automatic" is actually the default setting
demo.ridge <- linearRidge(y ~ x1 + x2 + x3 + x4, data=df, lambda="automatic")
coefficients(demo.ridge)
```

```
## (Intercept)          x1          x2          x3          x4
## 3.755564075 0.419326727 0.035301803 0.495563414 0.005749006
```

```
demo.ridge$lambda
```

```
## [1] 0.005617822 0.009013389 0.006462562
```

We may compare the predictions we get from this to the predictions we'd from dropping, say, X_2 (Figure 6). One substantial advantage of ridge regression is that we don't have to make any decisions about which variables to remove, but can match (to extremely high accuracy) what we'd get after dropping variables.

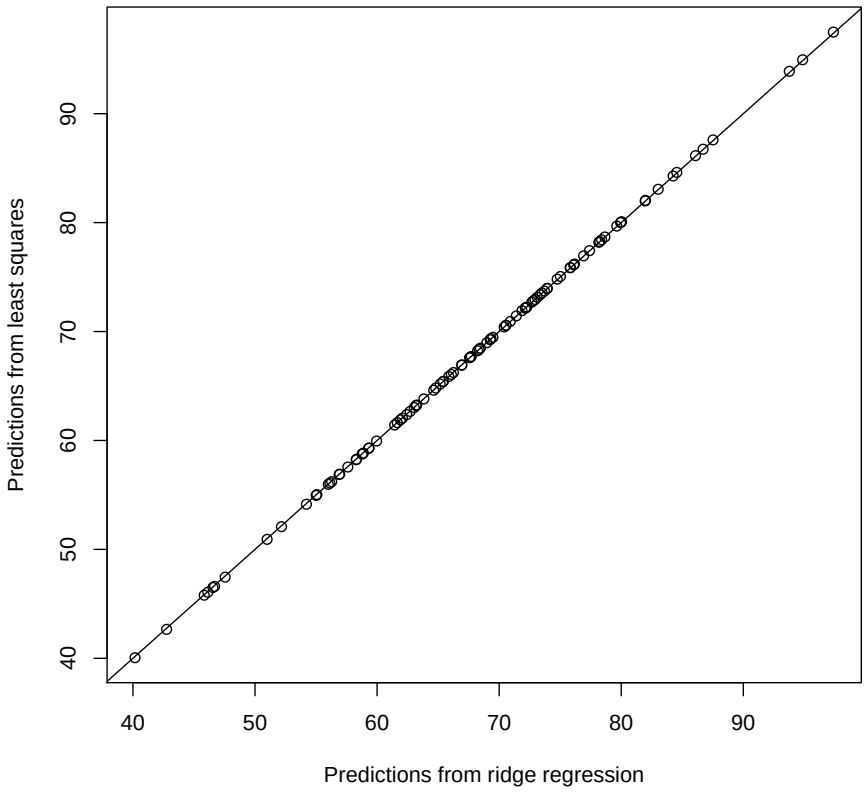


FIGURE 6: *Comparison of fitted values from an ordinary least squares regression where we drop X_2 from our running example (vertical axis) against fitted values from a ridge regression on all variables (horizontal axis); the two sets of numbers are not* exactly *equal, though they are close.*

6 Further Reading

Ridge regression, by that name, goes back to Hoerl and Kennard (1970). Essentially the same idea was introduced some years earlier by the great Soviet mathematician A. N. Tikhonov in a series of papers about "regularizing ill-posed optimization problems", i.e., adding penalties to optimization problems to make a solution unique, or to make the solution more stable. For this reason, ridge regression is sometimes also called "Tikhonov regularization" of linear least squares[16].

The use of principal components as a technique of dimension reduction goes back at least to Hotelling in the 1930s, or arguably to Karl Pearson around 1900. I have not been able to trace who first suggested regressing a response variable on the principal components of the predictors. Dhillon *et al.* (2013) establishes a surprising connection between regression on the principal components and ridge regression.

On the use of Lagrange multipliers to enforce constraints on optimization problems, and the general equivalence between penalized and constrained optimization, see Klein (2001), or Shalizi (forthcoming, §E.3).

For high-dimensional regression in general, the opening chapters of Bühlmann and van de Geer (2011) are very good. Bühlmann (2014); Wainwright (2014) may be more accessible review articles on basically the same topics.

For a representative example of the idea that big data "makes theory obsolete", see Anderson (2008); for a reply from someone who actually understands machine learning and high-dimensional statistics, see `http://earningmyturns.blogspot.com/2008/06/end-of-theory-data-deluge-makes.html`.

References

Anderson, Chris (June 2008). "The End of Theory: The Data Deluge Makes the Scientific Method Obsolete." *Wired*, **16(17)**. URL `http://www.wired.com/2008/06/pb-theory/`.

Bühlmann, Peter (2014). "High-Dimensional Statistics with a View Toward Applications in Biology." *Annual Review of Statistics and Its Applications*, **1**: 255–278. doi:10.1146/annurev-statistics-022513-115545.

Bühlmann, Peter and Sara van de Geer (2011). *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Berlin: Springer-Verlag.

Cule, Erika (2014). *`ridge`: Ridge Regression with automatic selection of the penalty parameter*. URL `http://CRAN.R-project.org/package=ridge`. R package version 2.1-3.

Dhillon, Paramveer S., Dean P. Foster, Sham M. Kakade and Lyle H. Ungar (2013). "A Risk Comparison of Ordinary Least Squares vs Ridge Regression."

[16] There are a large number of variant transliterations of "Tikhonov".

Journal of Machine Lerning Research, **14**: 1505–1511. URL http://jmlr.org/papers/v14/dhillon13a.html.

Hoerl, Arthur E. and Robert W. Kennard (1970). "Ridge Regression: Biased Estimation for Nonorthogonal Problems." *Technometrics*, **12**. URL http://www.jstor.org/pss/1267351.

Klein, Dan (2001). "Lagrange Multipliers without Permanent Scarring." Online tutorial. URL http://dbpubs.stanford.edu:8091/~klein/lagrange-multipliers.pdf.

Shalizi, Cosma Rohilla (forthcoming). *Advanced Data Analysis from an Elemntary Point of View*. Cambridge, England: Cambridge University Press. URL http://www.stat.cmu/~cshalizi/ADAfaEPoV.

Wainwright, Martin J. (2014). "Structured Regularizers for High-Dimensional Problems: Statistical and Computational Issues." *Annal Review of Statistics and Its Applications*, **1**: 233–253. doi:10.1146/annurev-statistics-022513-115643.