Chapter 9

Scale Changes, Principal Components and Collinearity

9.1 Changes of Scale

Suppose we re-express x_i as $\frac{x_i+a}{b}$. We might want to do this because

1. Predictors of similar magnitude are easier to compare. $\hat{\beta} = 3.51$ is easier to parse than $\hat{\beta} = 0.000000351$.

2. A change of units might aid interpretability.

3. Numerical stability is enhanced when all the predictors are on a similar scale.

Rescaling x_i leaves the t and F tests and $\hat{\sigma}^2$ and R^2 unchanged and $\hat{\beta}_i \rightarrow b\hat{\beta}_i$.
Rescaling y in the same way leaves the t and F tests and R^2 unchanged but $\hat{\sigma}$ and $\hat{\beta}$ will rescaled by b.
To demonstrate this, we use same old model:

```
> g <- lm(sr ~ pop15+pop75+dpi+ddpi,savings)
> summary(g)
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 28.566087   7.354516    3.88  0.00033
pop15       -0.461193   0.144642   -3.19  0.00260
pop75       -1.691498   1.083599   -1.56  0.12553
dpi         -0.000337   0.000931   -0.36  0.71917
ddpi         0.409695   0.196197    2.09  0.04247

Residual standard error: 3.8 on 45 degrees of freedom
Multiple R-Squared: 0.338,      Adjusted R-squared: 0.28
F-statistic: 5.76 on 4 and 45 degrees of freedom,       p-value: 0.00079
```

The coefficient for income is rather small - let's measure income in thousands of dollars instead and refit:

```
> g <- lm(sr ~ pop15+pop75+I(dpi/1000)+ddpi,savings)
> summary(g)
Coefficients:
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   28.566      7.355    3.88  0.00033
pop15         -0.461      0.145   -3.19  0.00260
pop75         -1.691      1.084   -1.56  0.12553
I(dpi/1000)   -0.337      0.931   -0.36  0.71917
ddpi           0.410      0.196    2.09  0.04247

Residual standard error: 3.8 on 45 degrees of freedom
Multiple R-Squared: 0.338,      Adjusted R-squared: 0.28
F-statistic: 5.76 on 4 and 45 degrees of freedom,       p-value: 0.00079
```

What changed and what stayed the same?

One rather thorough approach to scaling is to convert all the variables to standard units (mean 0 and variance 1) using the `scale()` command:

```
> scsav <- data.frame(scale(savings))
> g <- lm(sr ˜ ., scsav)
> summary(g)
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.0e-16     0.1200 3.3e-15   1.0000
pop15        -0.9420     0.2954   -3.19   0.0026
pop75        -0.4873     0.3122   -1.56   0.1255
dpi          -0.0745     0.2059   -0.36   0.7192
ddpi          0.2624     0.1257    2.09   0.0425

Residual standard error: 0.849 on 45 degrees of freedom
Multiple R-Squared: 0.338,      Adjusted R-squared: 0.28
F-statistic: 5.76 on 4 and 45 degrees of freedom,       p-value: 0.00079
```

As may be seen, the intercept is zero. This is because the regression plane always runs through the point of the averages which because of the centering is now at the origin. Such scaling has the advantage of putting all the predictors and the response on a comparable scale, which makes comparisons simpler. It also allows the coefficients to be viewed as kind of partial correlation — the values will always be between -1 and 1. It also avoids some numerical problems that can arise when variables are of very different scales. The downside of this scaling is that the regression coefficients now represent the effect of a one standard unit increase in the predictor on the response in standard units — this might not always be easy to interpret.

9.2 Principal Components

Recall that if the X matrix is orthogonal then testing and interpretation are greatly simplified. One purpose for principal components is to transform the X to orthogonality. For example, consider the case with two predictors depicted in Figure 9.1.

The original predictors, x_1 and x_2, are clearly correlated and so the X-matrix will not be orthogonal. This will complicate the interpretation of the effects of x_1 and x_2 on the response. Suppose we rotate the coordinate axes so that in the new system, the predictors are orthogonal. Furthermore, suppose we make the rotation so that the first axis lies in the direction of the greatest variation in the data, the second in the

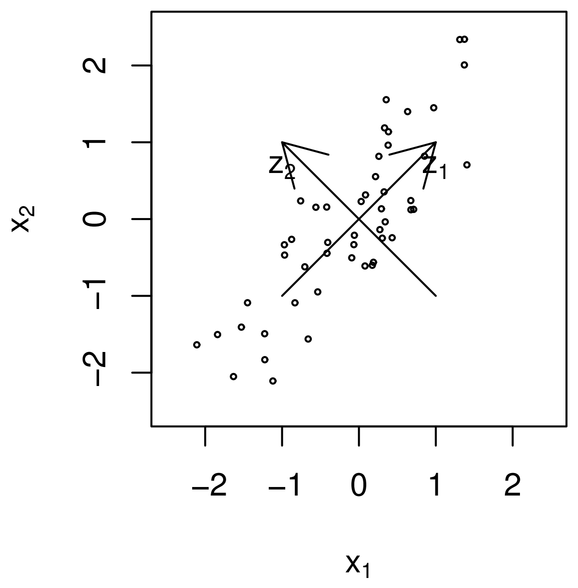

Figure 9.1: Original predictors are x_1 and x_2, principal components are z_1 and z_2

second greatest direction of variation in those dimensions remaining and so on. These rotated directions, z_1 and z_2 in our two predictor example, are simply linear combinations of the original predictors. This is the geometrical description of principal components. We now indicate how these directions may be calculated.

We wish to find a rotation $p \times p$ matrix U such that

$$Z = XU$$

and $Z^T Z = \text{diag}(\lambda_1, \ldots, \lambda_p)$ and $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p \geq 0$. Zero eigenvalues indicate non-identifiability. Since

$$Z^T Z = U^T X^T X U$$

the eigenvalues of $X^T X$ are $\lambda_1, \ldots, \lambda_p$ and the eigenvectors of $X^T X$ are the columns of U. The columns of Z are called the principal components and these are orthogonal to each other. λ_i is the variance of Z_i.

Another way of looking at it is to try to find the linear combinations of X which have the maximum variation. We find the u_1 such that var $(u_1^T X)$ is maximized subject to $u_1^T u_1 = 1$. Now find u_2 such that var $(u_2^T X)$ is maximized subject to $u_1^T u_2 = 0$ and $u_2^T u_2 = 1$. We keep finding directions of greatest variation orthogonal to those directions we have already found.

Ideally, only a few eigenvalues will be large so that almost all the variation in X will be representable by those first few principal components.

Principal components can be effective in some situations but

1. The principal components are linear combinations of the predictors — little is gained if these are not interpretable. Generally the predictors have to be measurements of comparable quantities for interpretation to be possible

2. Principal components does not use the response. It's possible that a lesser principal component is actually very important in predicting the response.

3. There are variations which involve not using the intercept in $X^T X$ or using the correlation matrix of the predictors instead of $X^T X$.

We use the Longley data for this example: First we compute the eigendecomposition for $X^T X$:

```
> data(longley)
> x <- as.matrix(longley[,-7])
> e <- eigen(t(x) %*% x)
```

Look at the eigenvalues:

```
> e$values
[1] 6.6653e+07 2.0907e+05 1.0536e+05 1.8040e+04 2.4557e+01 2.0151e+00
```

Look at the relative size - the first is big. Consider the first eigenvector (column) below:

```
> dimnames(e$vectors)[[2]] <- paste("EV",1:6)
> round(e$vec,3)
         EV 1   EV 2   EV 3   EV 4   EV 5   EV 6
GNP def 0.050 -0.070 -0.034  0.043 -0.957 -0.273
GNP     0.191 -0.725 -0.343  0.554  0.075  0.087
Unem    0.157 -0.622  0.564 -0.521  0.007  0.011
Armed   0.128 -0.104 -0.746 -0.645  0.012  0.000
Popn    0.058 -0.038 -0.011  0.036  0.281 -0.956
Year    0.957  0.266  0.078  0.057  0.015  0.053
```

The first eigenvector is dominated by year. Now examining the X-matrix. What are the scales of the variables?

```
> x
     GNP deflator     GNP Unemployed Armed Forces Population Year
1947         83.0 234.289      235.6        159.0    107.608 1947
1948         88.5 259.426      232.5        145.6    108.632 1948
1949         88.2 258.054      368.2        161.6    109.773 1949
1950         89.5 284.599      335.1        165.0    110.929 1950
1951         96.2 328.975      209.9        309.9    112.075 1951
1952         98.1 346.999      193.2        359.4    113.270 1952
1953         99.0 365.385      187.0        354.7    115.094 1953
1954        100.0 363.112      357.8        335.0    116.219 1954
1955        101.2 397.469      290.4        304.8    117.388 1955
1956        104.6 419.180      282.2        285.7    118.734 1956
1957        108.4 442.769      293.6        279.8    120.445 1957
1958        110.8 444.546      468.1        263.7    121.950 1958
1959        112.6 482.704      381.3        255.2    123.366 1959
1960        114.2 502.601      393.1        251.4    125.368 1960
1961        115.7 518.173      480.6        257.2    127.852 1961
1962        116.9 554.894      400.7        282.7    130.081 1962
```

We see that the variables have different scales. It might make more sense to standardize the predictors before trying principal components. This is equivalent to doing principal components on the correlation matrix:

```
> e <- eigen(cor(x))
> e$values
[1] 4.60337710 1.17534050 0.20342537 0.01492826 0.00255207 0.00037671
> dimnames(e$vectors) <- list(c("GNP def","GNP","Unem","Armed","Popn",
  "Year"),paste("EV",1:6))
> round(e$vec,3)
         EV 1   EV 2   EV 3   EV 4   EV 5   EV 6
GNP def 0.462  0.058 -0.149  0.793  0.338  0.135
GNP     0.462  0.053 -0.278 -0.122 -0.150 -0.818
Unem    0.321 -0.596  0.728  0.008  0.009 -0.107
Armed   0.202  0.798  0.562 -0.077  0.024 -0.018
Popn    0.462 -0.046 -0.196 -0.590  0.549  0.312
Year    0.465  0.001 -0.128 -0.052 -0.750  0.450
```

One commonly used method of judging how many principal components are worth considering is the *scree plot* — see Figure 9.2, which is produced by

```
> plot(e$values,type="l",xlab="EV no.")
```

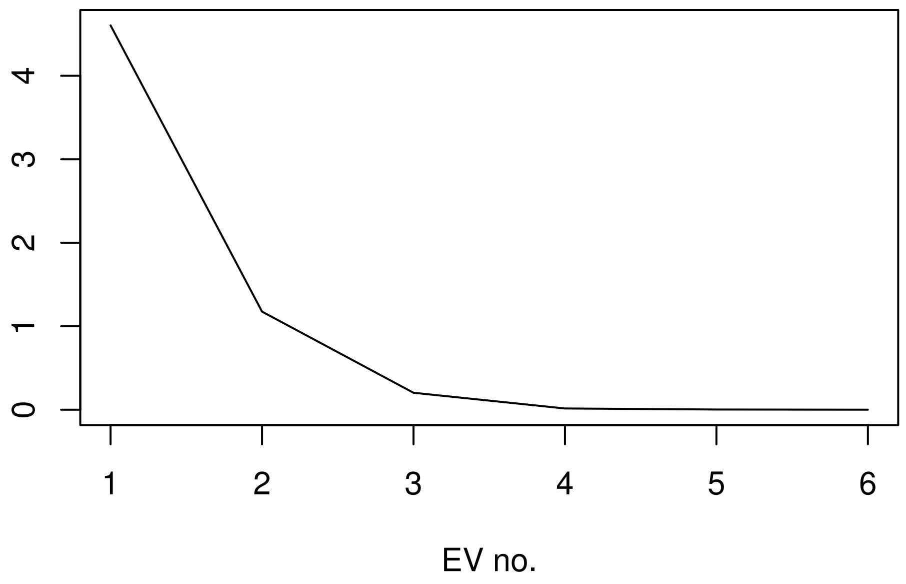

Figure 9.2: Scree plot for the principal components on the correlation of longley predictors

Often, these plots have a noticeable "elbow" — the point at which further eigenvalues are negligible in size compared to the earlier ones. Here the elbow is at 3 telling us that we need only consider 2 principal components.

One advantage of principal components is that it transforms the predictors to an orthogonal basis. To figure out the orthogonalized predictors for this data based on the eigendecomposition for the correlation matrix we must first standardize the data: The functions `scale()` does this:

```
> nx <- scale(x)
```

We can now create the orthogonalized predictors — the $Z = XU$ operation in our description above.

```
> enx <- nx %*% e$vec
```

and fit:

```
> g <- lm(longley$Emp ~ enx)
> summary(g)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  65.3170     0.0762  857.03  < 2e-16
enxEV 1       1.5651     0.0367   42.66  1.1e-11
enxEV 2       0.3918     0.0726    5.40  0.00043
enxEV 3      -1.8604     0.1745  -10.66  2.1e-06
enxEV 4       0.3573     0.6442    0.55  0.59267
enxEV 5      -6.1698     1.5581   -3.96  0.00331
enxEV 6       6.9634     4.0555    1.72  0.12011

Residual standard error: 0.305 on 9 degrees of freedom
Multiple R-Squared: 0.995,      Adjusted R-squared: 0.992
F-statistic:  330 on 6 and 9 degrees of freedom,        p-value: 4.98e-10
```

Notice that the p-values of the 4th and 6th eigenvectors are not significant while the the 5th is. Because the directions of the eigenvectors are set successively in the greatest remaining direction of variation in the predictors, it is natural that they be ordered in significance in predicting the response. However, there is no guarantee of this — we see here that the 5th eigenvectors is significant while the fourth is not even though there is about six times greater variation in the fourth direction than the fifth. In this example, it hardly matters since most of the variation is explained by the earlier values, but look out for this effect in other dataset in the first few eigenvalues.

We can take a look at the $(X^T X)^{-1}$ matrix:

```
> round(summary(g)$cov.unscaled,2)
            (Intercept) enxEV 1 enxEV 2 enxEV 3 enxEV 4 enxEV 5 enxEV 6
(Intercept)        0.06    0.00    0.00    0.00    0.00    0.00    0.00
enxEV 1            0.00    0.01    0.00    0.00    0.00    0.00    0.00
enxEV 2            0.00    0.00    0.06    0.00    0.00    0.00    0.00
enxEV 3            0.00    0.00    0.00    0.33    0.00    0.00    0.00
enxEV 4            0.00    0.00    0.00    0.00    4.47    0.00    0.00
enxEV 5            0.00    0.00    0.00    0.00    0.00   26.12    0.00
enxEV 6            0.00    0.00    0.00    0.00    0.00    0.00  176.97
```

Principal components are really only useful if we can interpret the meaning of the new linear combinations. Look back at the first eigenvector - this is roughly a linear combination of all the (standardized variables). Now plot each of the variables as they change over time — see Figure 9.2.

```
> par(mfrow=c(3,2))
> for(i in 1:6) plot(longley[,6],longley[,i],xlab="Year",
  ylab=names(longley)[i])
```

What do you notice? This suggests we identify the first principal component with a time trend effect. The second principal component is roughly a contrast between numbers unemployed and in the armed forces. Let's try fitting a regression with those two components:

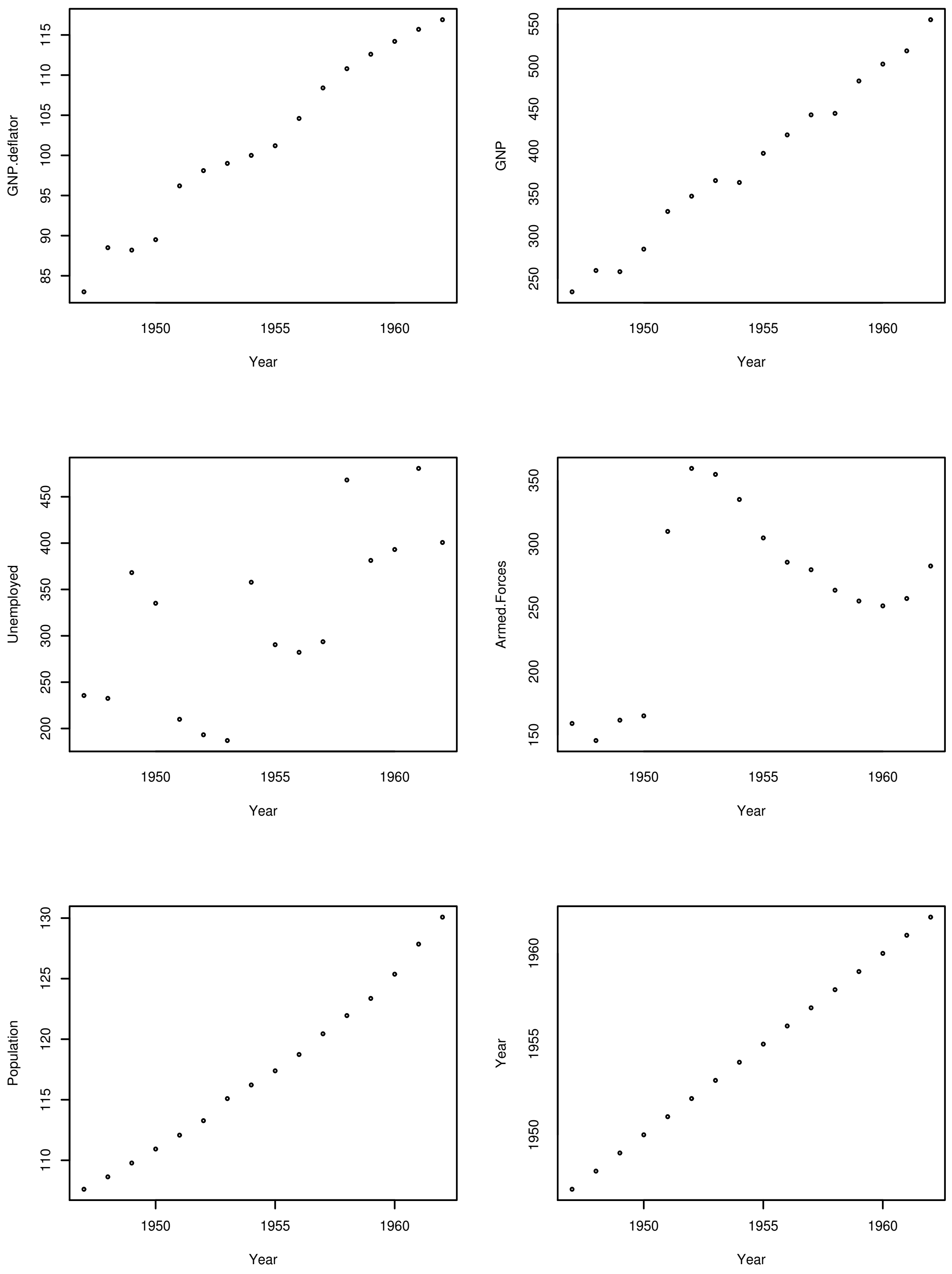

Figure 9.3: The Longley data

```
> summary(lm(Employed ˜ Year + I(Unemployed-Armed.Forces),longley))
Coefficients:
                               Estimate Std. Error t value Pr(>|t|)
(Intercept)                   -1.39e+03   7.89e+01   -17.6  1.8e-10
Year                           7.45e-01   4.04e-02    18.5  1.0e-10
I(Unemployed - Armed.Forces) -4.12e-03   1.53e-03    -2.7    0.018

Residual standard error: 0.718 on 13 degrees of freedom
Multiple R-Squared: 0.964,      Adjusted R-squared: 0.958
F-statistic:  173 on 2 and 13 degrees of freedom,       p-value: 4.29e-10
```

This approaches the fit of the full model and is easily interpretable. We could do more work on the other principal components.

This is illustrates a typical use of principal components for regression. Some intuition is required to form new variables as combinations of older ones. If it works, a simplified and interpretable model is obtained, but it doesn't always work out that way.

9.3 Partial Least Squares

Partial Least Squares is a method for relating a set of input variables $X_1,\dots,X_m$ and outputs $Y_1,\dots,Y_l$. PLS has some relationship to principal component regression (PCR). PCR regresses the response on the principal components of X while PLS finds the best orthogonal linear combinations of X for predicting Y.

We will consider only univariate PLS — that is to say $l = 1$ so that Y is scalar. This is the typical multiple regression setting. We will attempt to find models of the form

$$\hat{y} = \beta_1 T_1 + \ldots + \beta_p T_p$$

where T_k is a linear combination of the X's. See Figure 9.4

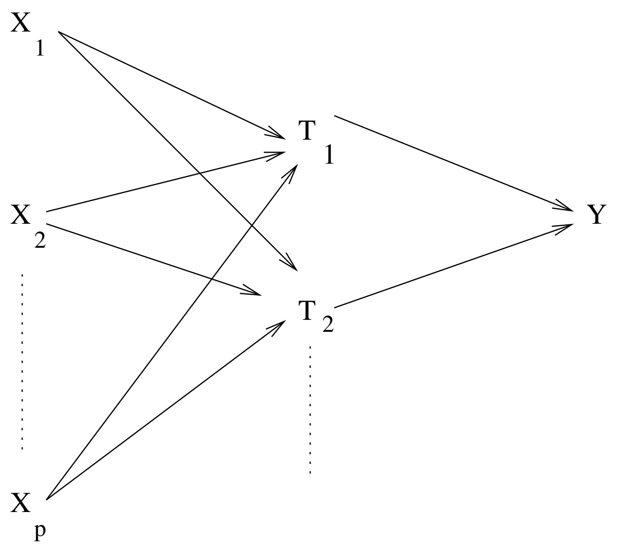

Figure 9.4: Schematic representation of Partial Least Squares

We'll assume that all variables have been centered to have mean 0 — this means that our intercept terms will always be zero. Here is the algorithm for determining the T's.

1. Regress Y on each X_i in turn to get b_{1i}.
2. Form
$$T_1 = \sum_{i=1}^{m} w_{1i} b_{1i} X_{1i}$$
where the weights w_{1i} sum to one.
3. Regress Y on T_1 and each X_i on T_1. The residuals from these regressions have the effect of T_1 removed. Replace Y and each X_i by the residuals of each corresponding regression.
4. Go back to step one updating the index.

There are various choices for the weighting scheme:

1. Set $w_{ij} = 1/m$ giving each predictor equal weight.
2. Set $w_{ij} \propto \text{var}X_j$. This is the most common choice. The variances of the b_{ij} are then inversely proportional to $\text{var}X_j$ which does make some sense.

The algorithm ensures that the components T_i are uncorrelated just as the principal components are uncorrelated. This means that $\hat{\beta}_i$ will not change as more components are added to or subtracted from the model.

For this example, we again use the Longley data. We will not need intercept terms in any of the regressions because of the centering.

```
> x <- as.matrix(longley[,-7])
> y <- longley$Emp
> cx <- sweep(x,2,apply(x,2,mean))
> cy <- y -mean(y)
```

Now do the PCR using a more direct method than we used above:

```
> library(mva)
> ex <- princomp(cx)
> g <- lm(cy ~ ex$scores -1)
> summary(g)
Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
ex$scoresComp.1  0.025095   0.000603   41.65  1.5e-12
ex$scoresComp.2  0.014038   0.000888   15.82  2.1e-08
ex$scoresComp.3  0.029991   0.002152   13.94  7.1e-08
ex$scoresComp.4 -0.061765   0.058332   -1.06   0.3146
ex$scoresComp.5 -0.489877   0.200645   -2.44   0.0348
ex$scoresComp.6  1.762076   0.443363    3.97   0.0026

Residual standard error: 0.289 on 10 degrees of freedom
Multiple R-Squared: 0.995,      Adjusted R-squared: 0.993
F-statistic:  367 on 6 and 10 degrees of freedom,       p-value: 3.94e-11
```

Are the principal component scores ordered in terms of their importance in predicting the response? Now for later comparison, we have the regression on just first PC.

```
> g <- lm(cy ~ ex$scores[,1] -1)
> summary(g)
Coefficients:
               Estimate Std. Error t value Pr(>|t|)
ex$scores[, 1]   0.0251     0.0034    7.38  2.3e-06

Residual standard error: 1.63 on 15 degrees of freedom
Multiple R-Squared: 0.784,      Adjusted R-squared: 0.77
F-statistic: 54.5 on 1 and 15 degrees of freedom,       p-value: 2.28e-06
```

Now we compute the first component of the partial least squares regression:

```
> b1 <- numeric(6)
> for(i in 1:6){
+ b1[i] <- crossprod(cx[,i],cy)/crossprod(cx[,i],cx[,i])
+ }
> b1
[1] 0.315966 0.034752 0.018885 0.023078 0.484878 0.716512
> ncx <- sweep(cx,2,b1,"*")
> t1 <- apply(ncx,1,mean)
```

Here we have a chosen an equal weighting for the variables. Now see how well this component predicts the response:

```
> gpls1 <- lm(cy ~ t1 -1)
> summary(gpls1)
Coefficients:
   Estimate Std. Error t value Pr(>|t|)
t1   1.3108     0.0959    13.7  7.1e-10

Residual standard error: 0.957 on 15 degrees of freedom
Multiple R-Squared: 0.926,      Adjusted R-squared: 0.921
F-statistic:  187 on 1 and 15 degrees of freedom,       p-value: 7.11e-10
```

Compare this to the result above for one principal component.

An alternative weighting scheme assigns the weights proportional to the variances of the variables:

```
> varx <- apply(cx,2,var)
> vncx <- sweep(ncx,2,varx,"*")
> t1a <- apply(vncx,1,sum)/sum(varx)
> gpls1a <- lm(cy ~ t1a -1)
> summary(gpls1a)
Coefficients:
    Estimate Std. Error t value Pr(>|t|)
t1a    1.605      0.154    10.4  2.8e-08
```

```
Residual standard error: 1.22 on 15 degrees of freedom
Multiple R-Squared: 0.879,      Adjusted R-squared: 0.871
F-statistic:  109 on 1 and 15 degrees of freedom,       p-value: 2.81e-08
```

Compare this to the other output. Now we compute the second component of the PLS. We need to regress out the effect of the first component and then use the same computational method as above.

```
> cx2 <- matrix(0,16,6)
> for(i in 1:6){
+ cx2[,i] <- lm(cx[,i] ˜ t1-1)$res
+ }
> cy2 <- lm(cy ˜ t1 -1)$res
> b2 <- numeric(6)
> for(i in 1:6){
+ b2[i] <- crossprod(cx2[,i],cy2)/crossprod(cx2[,i],cx2[,i])
+ }
> ncx2 <- sweep(cx2,2,b2,"*")
> t2 <- apply(ncx2,1,mean)
```

Notice the correlation of the components:

```
> cor(t1,t2)
[1] 9.0843e-19
```

Now add t2 to the regression:

```
> gpls2 <- lm(cy ˜ t1+t2 -1)
> summary(gpls2)
Coefficients:
   Estimate Std. Error t value Pr(>|t|)
t1   1.3108     0.0749   17.49  6.5e-11
t2  10.9309     3.3658    3.25   0.0058

Residual standard error: 0.748 on 14 degrees of freedom
Multiple R-Squared: 0.958,      Adjusted R-squared: 0.952
F-statistic:  158 on 2 and 14 degrees of freedom,       p-value: 2.44e-10
```

Compare the coefficient of t1 with that above. Now compare this fit to the two component PCR.

```
> g <- lm(cy ˜ ex$scores[,1:2] -1)
> summary(g)
Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
ex$scores[, 1:2]Comp.1  0.02510    0.00243   10.34  6.2e-08
ex$scores[, 1:2]Comp.2  0.01404    0.00358    3.93   0.0015

Residual standard error: 1.16 on 14 degrees of freedom
Multiple R-Squared: 0.897,      Adjusted R-squared: 0.883
F-statistic: 61.2 on 2 and 14 degrees of freedom,       p-value: 1.2e-07
```

Which one is superior in explaining y?
Notes:

- The tricky part is choosing how many components are required. Crossvalidation is a possible way of selecting the number of components.
- There are other faster versions of the algorithm described above but these generally provide less insight into the method.
- PLS has been criticized as an algorithm that solves no well-defined modeling problem.
- PLS has the biggest advantage over ordinary least squares and PCR when there are large numbers of variables relative to the number of case. It does not even require that $n \geq p$.

PCR and PLS compared

PCR attempts to find linear combinations of the predictors that explain most of the variation in these predictors using just a few components. The purpose is dimension reduction. Because the principal components can be linear combinations of all the predictors, the number of variables used is not always reduced. Because the principal components are selected using only the X-matrix and not the response, there is no definite guarantee that the PCR will predict the response particularly well although this often happens. If it happens that we can interpret the principal components in a meaningful way, we may achieve a much simpler explanation of the response. Thus PCR is geared more towards explanation than prediction.

In contrast, PLS finds linear combinations of the predictors that best explain the response. It is most effective when ther are large numbers of variables to be considered. If successful, the variablity of prediction is substantially reduced. On the other hand, PLS is virtually useless for explanation purposes.

9.4 Collinearity

If X^TX is singular, i.e. some predictors are linear combinations of others, we have (exact) collinearity and there is no unique least squares estimate of β. If X^TX is close to singular, we have (approximate) collinearity or multicollinearity (some just call it collinearity). This causes serious problems with the estimation of β and associated quantities as well as the interpretation. Collinearity can be detected in several ways:

1. Examination of the correlation matrix of the predictors will reveal large *pairwise* collinearities.
2. A regression of x_i on all other predictors gives R_i^2. Repeat for all predictors. R_i^2 close to one indicates a problem — the offending linear combination may be found.
3. Examine the eigenvalues of X^TX - small eigenvalues indicate a problem. The condition number is defined as

$$\kappa = \sqrt{\frac{\lambda_1}{\lambda_p}}$$

where $\kappa \geq 30$ is considered large. κ is called the condition number. Other condition numbers, $\sqrt{\lambda_1/\lambda_i}$ are also worth considering because they indicate whether more than just one independent linear combination is to blame.

Collinearity makes some of the parameters hard to estimate. Define

$$S_{x_jx_j} = \sum_i (x_{ij} - \bar{x}_j)^2$$

then

$$\text{var } \hat{\beta}_j = \sigma^2 \left(\frac{1}{1 - R_j^2} \right) \frac{1}{S_{x_j x_j}}$$

We can see that if x_j doesn't vary much then the variance of $\hat{\beta}_j$ will be large. As an aside, the variance of the first principal component is maximized and so the variance of the corresponding regression coefficient will tend to be small. Another consequence of this equation is that it tells us what designs will minimize the variance of the regression coefficients if we have the ability to place the X. Orthogonality means that $R_j^2 = 0$ which minimizes the variance. Also we can maximize $S_{x_j x_j}$ by spreading X as much as possible. The maximum is attained by placing half the points at the minimum practical value and half at the maximum. Unfortunately, this design assumes the linearity of the effect and would make it impossible to check for any curvature. So, in practice, most would put some design points in the middle of the range to allow checking of the fit.

If R_j^2 is close to one then the *variance inflation factor* $\frac{1}{1-R_j^2}$ will be large. For orthogonal designs and principal components, $R_j^2 = 0$, so in these case, we see that the regression coefficient estimation suffers no additional penalty in terms of precision.

Collinearity leads to

1. imprecise estimates of β — the signs of the coefficients may be misleading.

2. t-tests which fail to reveal significant factors

3. missing importance of predictors

The Longley dataset is a good example of collinearity:

```
> g <- lm(Employed ˜ ., longley)
> summary(g)
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -3.48e+03   8.90e+02   -3.91  0.00356
GNP.deflator  1.51e-02   8.49e-02    0.18  0.86314
GNP          -3.58e-02   3.35e-02   -1.07  0.31268
Unemployed   -2.02e-02   4.88e-03   -4.14  0.00254
Armed.Forces -1.03e-02   2.14e-03   -4.82  0.00094
Population   -5.11e-02   2.26e-01   -0.23  0.82621
Year          1.83e+00   4.55e-01    4.02  0.00304

Residual standard error: 0.305 on 9 degrees of freedom
Multiple R-Squared: 0.995,      Adjusted R-squared: 0.992
F-statistic:  330 on 6 and 9 degrees of freedom,       p-value: 4.98e-10
```

Recall that the response is number employed. Three of the predictors have large p-values but all are variables that might be expected to affect the response. Why aren't they significant? Check the correlation matrix first (rounding to 3 digits for convenience)

```
> round(cor(longley[,-7]),3)
             GNP deflator   GNP Unemployed Armed Forces Population  Year
GNP deflator        1.000 0.992      0.621        0.465      0.979 0.991
```

```
         GNP        0.992 1.000       0.604          0.446       0.991 0.995
  Unemployed        0.621 0.604       1.000         -0.177       0.687 0.668
Armed Forces        0.465 0.446      -0.177          1.000       0.364 0.417
  Population        0.979 0.991       0.687          0.364       1.000 0.994
        Year        0.991 0.995       0.668          0.417       0.994 1.000
```

There are several large pairwise correlations. Now we check the eigendecomposition:

```
> x <- as.matrix(longley[,-7])
> e <- eigen(t(x) %*% x)
> e$val
[1] 6.6653e+07 2.0907e+05 1.0536e+05 1.8040e+04 2.4557e+01 2.0151e+00
> sqrt(e$val[1]/e$val)
[1]    1.000   17.855   25.153   60.785 1647.478 5751.216
```

There is a wide range in the eigenvalues and several condition numbers are large. This means that problems are being caused by more than just one linear combination. Now check out the variance inflation factors. For the first variable this is

```
> summary(lm(x[,1] ~ x[,-1]))$r.squared
[1] 0.99262
> 1/(1-0.99262)
[1] 135.5
```

which is large - the VIF for orthogonal predictors is 1. Now we compute all the VIF's in one go:

```
> vif(x)
[1]  135.5324 1788.5135   33.6189    3.5889  399.1510  758.9806
```

There's definitely a lot of variance inflation! For example, we can interpret $\sqrt{(}1788) \approx 42$ as telling us that the standard error for GNP is 42 times larger than it would have been without collinearity. How can we get rid of this problem? One way is to throw out some of the variables. Examine the full correlation matrix above. Notice that variables 3 and 4 do not have extremely large pairwise correlations with the other variables so we should keep them and focus on the others for candidates for removal:

```
> cor(x[,-c(3,4)])
             GNP.deflator     GNP Population    Year
GNP.deflator      1.00000 0.99159    0.97916 0.99115
GNP               0.99159 1.00000    0.99109 0.99527
Population        0.97916 0.99109    1.00000 0.99395
Year              0.99115 0.99527    0.99395 1.00000
```

These four variables are strongly correlated with each other - any one of them could do the job of representing the other. We pick year arbitrarily:

```
> summary(lm(Employed ~ Armed.Forces + Unemployed + Year,longley))
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.80e+03   6.86e+01  -26.18  5.9e-12
```

```
Armed.Forces -7.72e-03   1.84e-03   -4.20   0.0012
Unemployed   -1.47e-02   1.67e-03   -8.79  1.4e-06
Year          9.56e-01   3.55e-02   26.92  4.2e-12

Residual standard error: 0.332 on 12 degrees of freedom
Multiple R-Squared: 0.993,      Adjusted R-squared: 0.991
F-statistic:  555 on 3 and 12 degrees of freedom,       p-value: 3.92e-13
```

Comparing this with the original fit, we see that the fit is very similar but only three rather than six predictors are used.

One final point - extreme collinearity can cause problems in computing the estimates - look what happens when we use the direct formula for $\hat{\beta}$.

```
> x <- as.matrix(cbind(1,longley[,-7]))
> solve(t(x) %*% x) %*% t(x) %*% longley[,7]
Error: singular matrix 'a' in solve
```

R , like most statistical packages, uses a more numerically stable method for computing the estimates in `lm()` . Something more like this:

```
> solve(t(x) %*% x , t(x) %*% longley$Emp, tol = 1e-12)
            [,1]
[1,] -3.4822e+03
[2,]  1.5061e-02
[3,] -3.5818e-02
[4,] -2.0202e-02
[5,] -1.0332e-02
[6,] -5.1110e-02
[7,]  1.8291e+00
```

Collinearity can be interpreted geometrically. Imagine a table — as two diagonally opposite legs are moved closer together, the table becomes increasing unstable.

The effect of collinearity on prediction depends on where the prediction is to be made. The greater the distance from the observed data, the more unstable the prediction. Distance needs to be considered in a Mahalanobis sense rather than Euclidean.

One cure for collinearity is amputation — too many variables are trying to do the same job of explaining the response. When several variables which are highly correlated are each associated with the response, we have to take care that we don't conclude that the variables we drop have nothing to do with the response.

9.5 Ridge Regression

Ridge regression makes the assumption that the regression coefficients (after normalization) are not likely to be very large. It is appropriate for use when the design matrix is collinear and the usual least squares estimates of β appear to be unstable.

Suppose that the predictors have been centered by their means and scaled by their standard deviations and that the response has been centered. The ridge regression estimates of β are then given by

$$\hat{\beta} = (X^T X + \lambda I)^{-1} X^T y$$

The ridge constant λ is usually selected from the range $[0,1]$.

The use of ridge regression can be motivated in two ways. Suppose we take a Bayesian point of view and put a prior (multivariate normal) distribution on β that expresses the belief that smaller values of β are more likely than larger ones. Large values of λ correspond to a belief that the β are really quite small whereas smaller values of λ correspond to a more relaxed belief about β. This is illustrated in Figure 9.5.

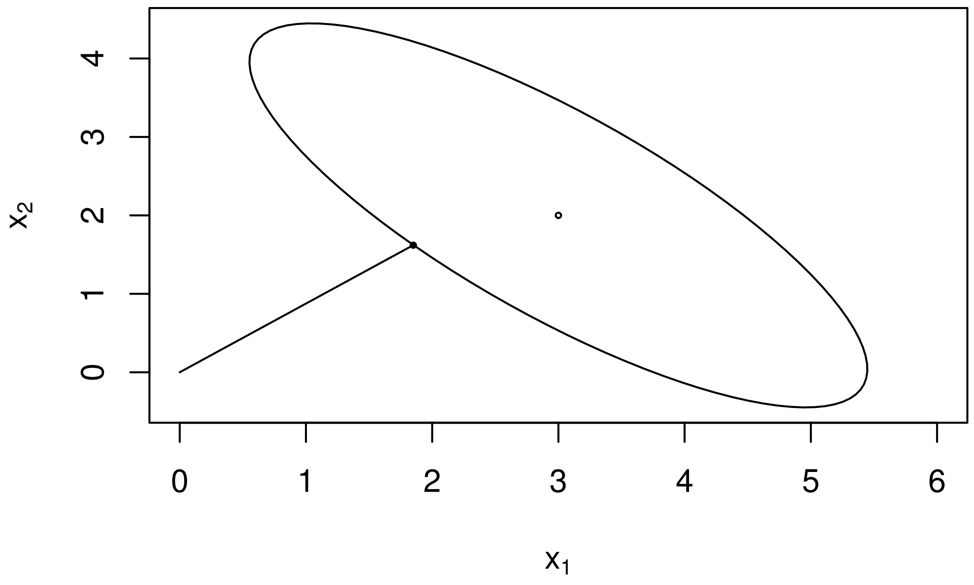

Figure 9.5: Ridge regression illustrated. The least squares estimate is at the center of the ellipse while the ridge regression is the point on the ellipse closest to the origin. The ellipse is a contour of equal density of the posterior probability, which in this case will be comparable to a confidence ellipse. λ controls the size of the ellipse - the larger λ is, the larger the ellipse will be

Another way of looking at is to suppose we place to some upper bound on $\beta^T\beta$ and then compute the least squares estimate of β subject to this restriction. Use of Lagrange multipliers leads to ridge regression. The choice of λ corresponds to the choice of upper bound in this formulation.

λ may be chosen by automatic methods but it is probably safest to plot the values of $\hat{\beta}$ as a function of λ. You should pick the smallest value of λ that produces stable estimates of β.

We demonstrate the method on the Longley data. $\lambda = 0$ corresponds to least squares while we see that as $\lambda \to \infty$, $\hat{\beta} \to 0$.

```
> library(MASS)
> gr <- lm.ridge(Employed ~ .,longley,lambda = seq(0,0.1,0.001))
> matplot(gr$lambda,t(gr$coef),type="l",xlab=expression(lambda),
                ylab=expression(hat(beta)))
> abline(h=0,lwd=2)
```

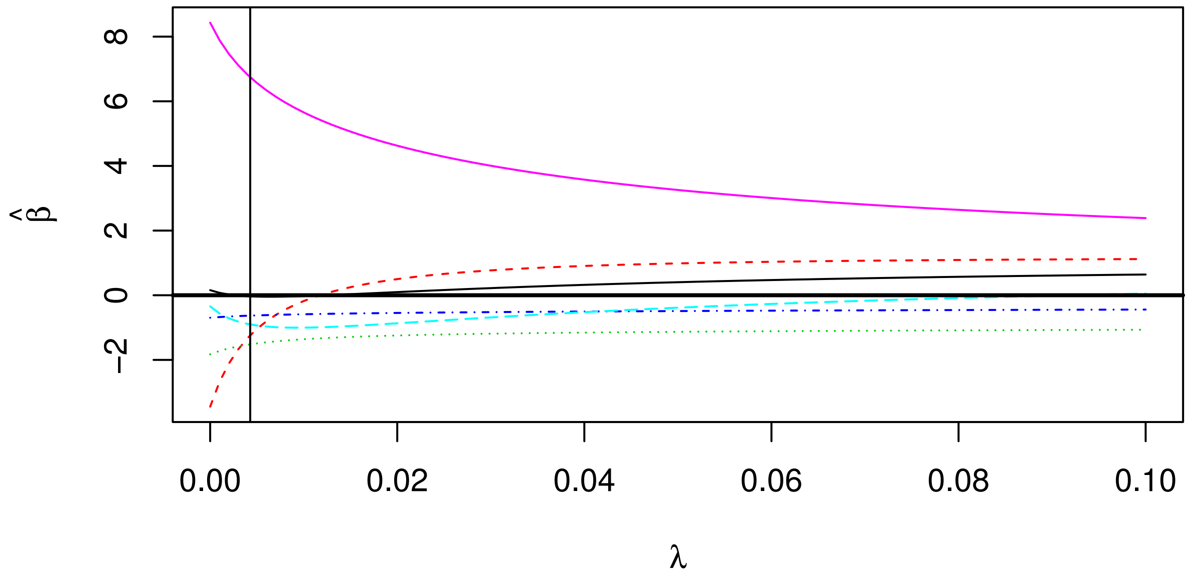

Figure 9.6: Ridge trace plot for the Longley data. The vertical line is the Hoerl-Kennard choice of λ. The topmost curve represent the coefficient for year. The dashed line that starts well below zero but ends above is for GNP.

The ridge trace plot is shown in Figure 9.5.

Various automatic selections for λ are available

```
> select(gr)
modified HKB estimator is 0.0042754
modified L-W estimator is 0.032295
smallest value of GCV  at 0.003
> abline(v=0.00428)
```

The Hoerl-Kennard (the originators of ridge regression) choice of λ has been shown on the plot but I would prefer a larger value of 0.03. For this choice of λ, the $\hat{\beta}$'s are

```
> gr$coef[,gr$lam == 0.03]
GNP.deflator        GNP   Unemployed Armed.Forces   Population      Year
     0.22005    0.76936     -1.18941     -0.52234     -0.68618   4.00643
```

in contrast to the least squares estimates of

```
> gr$coef[,1]
GNP.deflator        GNP  Unemployed Armed.Forces  Population      Year
     0.15738  -3.44719    -1.82789     -0.69621    -0.34420   8.43197
```

Note that these values are based on centered and scaled predictors which explains the difference from previous fits. Consider the change in the coefficient for GNP. For the least squares fit, the effect of GNP is negative on the response - number of people employed. This is counter-intuitive since we'd expect the effect to be positive. The ridge estimate is positive which is more in line with what we'd expect.

Ridge regression estimates of coefficients are biased. Bias is undesirable but there are other considerations. The mean squared error can be decomposed in the following way:

$$E(\hat{\beta}-\beta)^2 = (E(\hat{\beta}-\beta))^2 + E(\hat{\beta}-E\hat{\beta})^2$$

Thus the mean-squared error of an estimate can be represented as the square of the bias plus the variance. Sometimes a large reduction in the variance may obtained at the price of an increase in the bias. If the MSE is reduced as a consequence then we may be willing to accept some bias. This is the trade-off that Ridge Regression makes - a reduction in variance at the price of an increase in bias. This is a common dilemma.