Chapter 3

Properties of coefficient estimates

In chapter 1 we described properties of the coefficient estimates, $\widehat{\boldsymbol{\beta}}$, in the Gaussian linear model

$$\mathcal{Y} \sim \mathcal{N}(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2\boldsymbol{I}_n).$$

The estimates are called the *least squares estimates* because they minimize the sum of squared residuals from the observed responses, $\boldsymbol{y}$. That is,

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} S(\boldsymbol{\beta}) = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2$$

3.1 Geometric Properties

Recall that $\operatorname{col}(\boldsymbol{X})$, the *column span* of the $n \times p$ model matrix $\boldsymbol{X}$ is a linear subspace of the response space, $\mathbb{R}^n$,

$$\operatorname{col}(\boldsymbol{X}) = \{\boldsymbol{X}\boldsymbol{\beta} : \boldsymbol{\beta} \in \mathbb{R}^p\}$$

The dimension of $\operatorname{col}(\boldsymbol{X})$ is $k = \operatorname{rank}(\boldsymbol{X})$ and the QR decomposition used in R uses column pivoting to ensure that the first k columns of $\boldsymbol{Q}$ are an orthonormal basis for $\operatorname{col}(\boldsymbol{X})$.

At the risk of some confusion, we will refer to these k columns as $\boldsymbol{Q}_1$ which is equivalent to our previous definition in the most common case of full column rank for $\boldsymbol{X}$.

The fitted values, $\widehat{\boldsymbol{y}}$, are the (orthogonal) projection of $\boldsymbol{y}$ onto $\operatorname{col}(\boldsymbol{X})$

$$\widehat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y} = \boldsymbol{Q}_1\boldsymbol{Q}_1'\boldsymbol{y}$$

where the "hat matrix", $\boldsymbol{H} = \boldsymbol{Q}_1\boldsymbol{Q}_1'$, is a projection matrix of rank

$$\operatorname{tr}(\boldsymbol{Q}_1\boldsymbol{Q}_1') = \operatorname{tr}(\boldsymbol{Q}_1'\boldsymbol{Q}_1) = \operatorname{tr}(\boldsymbol{I}_k) = k$$

The diagonal matrix, $\boldsymbol{D}$, in the singular value decomposition (sect. 1.4.3, p. 17), $\boldsymbol{X} = \boldsymbol{U}_1\boldsymbol{D}\boldsymbol{V}'$, has exactly $p - k$ values that are (effectively) zero and these will be in the last $p - k$ positions. (Recall that the singular values, which must be non-negative, are in decreasing order.) Thus the first k columns of $\boldsymbol{U}_1$ also form an orthonormal basis for $\operatorname{col}(\boldsymbol{X})$.

The residual at the parameter estimates, $\widehat{\boldsymbol{e}} = \boldsymbol{y} - \widehat{\boldsymbol{y}}$ is orthogonal to col($\boldsymbol{X}$). We can prove this by showing that $\widehat{\boldsymbol{e}}$ is orthogonal to the k columns of $\boldsymbol{Q}_1$ which form a basis for col($\boldsymbol{X}$).

$$\boldsymbol{Q}_1'\widehat{\boldsymbol{e}} = \boldsymbol{Q}_1'\left(\boldsymbol{y} - \widehat{\boldsymbol{y}}\right) = \boldsymbol{Q}_1'\left(\boldsymbol{I}_n - \boldsymbol{Q}_1\boldsymbol{Q}_1'\right)\boldsymbol{y} = \left(\boldsymbol{Q}_1' - \underbrace{\boldsymbol{Q}_1'\boldsymbol{Q}_1}_{\boldsymbol{I}_p}\boldsymbol{Q}_1'\right)\boldsymbol{y} = \boldsymbol{0}$$

This is also an obvious geometric property that to minimize the distance between a point on a hyperplane and a general point in the response space, $\arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2$, you use orthogonal projection of $\boldsymbol{y}$ onto col($\boldsymbol{X}$) which implies that the residual is orthogonal to col($\boldsymbol{X}$).

Often this relationship is characterized as the *normal equations*. The residual will be orthogonal to col($\boldsymbol{X}$) if it is orthogonal to all the columns of $\boldsymbol{X}$, which is to say

$$\boldsymbol{X}'\left(\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}\right) = \boldsymbol{0} \;\Rightarrow\; \left(\boldsymbol{X}'\boldsymbol{X}\right)\widehat{\boldsymbol{\beta}} = \boldsymbol{X}'\boldsymbol{y}$$

3.2 Calculus Approach

The function

$$\begin{aligned} S(\boldsymbol{\beta}) &= \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 \\ &= \left(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\right)'\left(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\right) \\ &= \boldsymbol{y}'\boldsymbol{y} - \boldsymbol{y}'\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{\beta}'\boldsymbol{X}'\boldsymbol{y} + \boldsymbol{\beta}'\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\beta} \\ &= \boldsymbol{y}'\boldsymbol{y} - 2\boldsymbol{\beta}'\boldsymbol{X}'\boldsymbol{y} + \boldsymbol{\beta}'\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\beta} \end{aligned}$$

is a real-valued function of the p-vector, $\boldsymbol{\beta}$, ($S : \mathbb{R}^p \to \mathbb{R}$), with gradient vector

$$\frac{d\,S}{d\boldsymbol{\beta}} = -2\boldsymbol{X}'\boldsymbol{y} + 2\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\beta}.$$

Thus a critical point, $\boldsymbol{\beta}_c$, at which the gradient is zero, satisfies

$$\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\beta}_c = \boldsymbol{X}'\boldsymbol{y}.$$

The Hessian matrix of $S(\boldsymbol{\beta})$,

$$\frac{d^2\,S}{d\boldsymbol{\beta}\,d\boldsymbol{\beta}'} = 2\boldsymbol{X}'\boldsymbol{X}$$

is positive semi-definite. If $\boldsymbol{X}$ is full rank then $\boldsymbol{X}'\boldsymbol{X}$ is positive definite and the critical point will be the minimizer of $S(\boldsymbol{\beta})$.

3.3 Algebraic Properties of $\widehat{\beta}$

1. $\widehat{\boldsymbol{\beta}}$ satisfies $\boldsymbol{X}'\boldsymbol{X}\widehat{\boldsymbol{\beta}} = \boldsymbol{X}'\boldsymbol{y}$ and minimizes $S(\boldsymbol{\beta}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2$
2. If $\boldsymbol{X}$ is of rank p, then $\widehat{\boldsymbol{\beta}}$ is unique, satisfying $\boldsymbol{X}'\boldsymbol{X}\widehat{\boldsymbol{\beta}} = \boldsymbol{X}'\boldsymbol{y}$ or, equivalently, $\boldsymbol{R}\widehat{\boldsymbol{\beta}} = \boldsymbol{Q}_1'\boldsymbol{y}$ for an invertible, upper-triangular $p \times p$ matrix $\boldsymbol{R}$.

3. If $rank(\boldsymbol{X}) < p$, $\boldsymbol{X}'\boldsymbol{X}\widehat{\boldsymbol{\beta}} = \boldsymbol{X}'\boldsymbol{y}$ has multiple solutions for $\widehat{\boldsymbol{\beta}}$ but $\widehat{\boldsymbol{y}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}}$ is the same for all such $\widehat{\boldsymbol{\beta}}$

Proof. To prove item 3: Suppose that $\widehat{\boldsymbol{\beta}}_1$ and $\widehat{\boldsymbol{\beta}}_2$ are such that $\boldsymbol{X}'\boldsymbol{X}\widehat{\boldsymbol{\beta}}_1 = \boldsymbol{X}'\boldsymbol{X}\widehat{\boldsymbol{\beta}}_1 = \boldsymbol{X}'\boldsymbol{y}$. Then

$$\boldsymbol{X}'\left(\boldsymbol{X}\widehat{\boldsymbol{\beta}}_1 - \boldsymbol{X}\widehat{\boldsymbol{\beta}}_2\right) = \boldsymbol{0}$$

which implies that

$$0 = (\widehat{\boldsymbol{\beta}}_1 - \widehat{\boldsymbol{\beta}}_2)\boldsymbol{X}'\boldsymbol{X}(\widehat{\boldsymbol{\beta}}_1 - \widehat{\boldsymbol{\beta}}_2) = \|\boldsymbol{X}\widehat{\boldsymbol{\beta}}_1 - \boldsymbol{X}\widehat{\boldsymbol{\beta}}_2\|^2 \Rightarrow \boldsymbol{X}\widehat{\boldsymbol{\beta}}_1 = \boldsymbol{X}\widehat{\boldsymbol{\beta}}_2$$

□

3.4 Rank deficient cases and the Moore-Penrose inverse

In practice a rank-deficient model matrix, $\boldsymbol{X}$, is handled by two methods

1. Don't create it in the first place, use $I - 1$ *contrasts* for a factor with I levels.

2. Use the pivoted QR decomposition that retains the original order of the columns except that columns whose diagonal elements in $\boldsymbol{R}$ would be effectively zero are moved to trailing positions.

After that the calculation procedes as in the full-rank case except that only the first $k = \text{rank}(\boldsymbol{X})$ columns are used in $\boldsymbol{Q}_1$ and $\boldsymbol{R}$ is taken as the $k \times k$ upper-left submatrix of the calculated $p \times p$ $\boldsymbol{R}$.

When discussion the singular value decomposition in Chap. 1, we mentioned the pseudo-inverse or *generalized inverse* of $\boldsymbol{X}$, written $\boldsymbol{X}^-$, which formally is called the *Moore-Penrose generalized inverse*. If $\text{rank}(\boldsymbol{X}) = k < p$ then there are $p - k$ singular values of zero (in practice, very close to zero). The SVD is

$$\boldsymbol{X} = \boldsymbol{U}_1\boldsymbol{D}\boldsymbol{V}' = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{D}}\boldsymbol{V}'$$

where $\tilde{\boldsymbol{U}}$ is the first k columns of $\boldsymbol{U}_1$ and $\tilde{\boldsymbol{D}}$ is the first k rows of $\boldsymbol{D}$. $\boldsymbol{D}^-$, the Moore-Penrose inverse of $\boldsymbol{D}$, is also a diagonal matrix with diagonal elements $1/d_{i,i}$, $i = 1, \ldots, k$ and zero thereafter. The Moore-Penrose inverse of $\tilde{\boldsymbol{D}}$ is the first k columns of $\boldsymbol{D}^-$. Finally, the Moore-Penrose generalized inverse of $\boldsymbol{X}$ is

$$\boldsymbol{X}^- = \boldsymbol{V}\boldsymbol{D}^-\boldsymbol{U}_1' = \boldsymbol{V}\tilde{\boldsymbol{D}}^-\tilde{\boldsymbol{U}}'$$

This is an interesting theoretical tool but in practice it is not necessary to form the SVD in order to solve rank-deficient least squares problems.

3.4.1 Properties of Generalized Inverses

Let $\boldsymbol{A}$ be an $n \times p$ matrix and $\boldsymbol{A}^-$ be its $p \times n$ pseudo-inverse. The conditions that $\boldsymbol{A}$ and $\boldsymbol{A}^-$ must satisfy are

1. $\boldsymbol{A}\boldsymbol{A}^-\boldsymbol{A} = \boldsymbol{A}$ (i.e. $\boldsymbol{A}\boldsymbol{A}^-$ maps the columns of $\boldsymbol{A}$ to themselves.)

2. $\boldsymbol{A}^-\boldsymbol{A}\boldsymbol{A}^- = \boldsymbol{A}^-$ (i.e. $\boldsymbol{A}^-\boldsymbol{A}$ maps the columns of $\boldsymbol{A}^-$ to themselves.)

3. Both $\boldsymbol{A}\boldsymbol{A}^-$ and $\boldsymbol{A}^-\boldsymbol{A}$ are symmetric

(It is easy to verify these conditions for our case of $\boldsymbol{X}^- = \boldsymbol{V}\boldsymbol{D}^-\boldsymbol{U}$ where $\boldsymbol{X}$ is $n \times p$ with $\text{rank}(\boldsymbol{X}) \leq p \leq n$. In fact, you will do so on a homework assignment.)

Let $\boldsymbol{H} = \boldsymbol{A}^-\boldsymbol{A}$ be the associated projection in $\mathbb{R}^p$. Then the condition $\boldsymbol{A}\boldsymbol{A}^-\boldsymbol{A} = \boldsymbol{A}$ implies

1. $\boldsymbol{H}$ is idempotent because $\boldsymbol{H}\boldsymbol{H} = \boldsymbol{A}^-\boldsymbol{A}\boldsymbol{A}^-\boldsymbol{A} = \boldsymbol{A}^-\boldsymbol{A} = \boldsymbol{H}$.

2. $\boldsymbol{A}\boldsymbol{H} = \boldsymbol{A}$ (just plug in the definition of $\boldsymbol{H}$) so $\text{rank}(\boldsymbol{A}) \leq \text{rank}(\boldsymbol{H})$. However, we also have $\text{rank}(\boldsymbol{H}) \leq \text{rank}(\boldsymbol{A})$ because $\boldsymbol{H} = \boldsymbol{A}^-\boldsymbol{A}$. Thus $\text{rank}(\boldsymbol{A}) = \text{rank}(\boldsymbol{H}) = \text{tr}(\boldsymbol{H})$

3. A general solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ is
$$\boldsymbol{x} = (\boldsymbol{H} - \boldsymbol{I}_p)\boldsymbol{z}$$
where $\boldsymbol{z}$ is any vector in $\mathbb{R}^p$
$$\begin{aligned}\boldsymbol{A}\boldsymbol{x} &= \boldsymbol{A}(\boldsymbol{H} - \boldsymbol{I}_p)\boldsymbol{z}\\ &= (\boldsymbol{A}\boldsymbol{H} - \boldsymbol{A})\boldsymbol{z}\\ &= (\boldsymbol{A} - \boldsymbol{A})\boldsymbol{z} = \boldsymbol{0}\end{aligned}$$

4. A general solution to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ is
$$\boldsymbol{x} = \boldsymbol{A}^-\boldsymbol{y} + (\boldsymbol{H} - \boldsymbol{I}_p)\boldsymbol{z}$$
$$\boldsymbol{A}\boldsymbol{A}^-\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}\boldsymbol{x} \;\Rightarrow\; \boldsymbol{A}\boldsymbol{A}^-\boldsymbol{y} = \boldsymbol{y}$$
$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{A}^-\boldsymbol{y} + \underbrace{\boldsymbol{A}(\boldsymbol{H} - \boldsymbol{I}_p)\boldsymbol{z}}_{0} \text{ and } \boldsymbol{A}\boldsymbol{A}^-\boldsymbol{y} = \boldsymbol{y} \;\Rightarrow\; \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$$

$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^-\boldsymbol{X}'\boldsymbol{y}$ is a particular least squares solution in the rank deficient case. The general solution is
$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^-\boldsymbol{X}'\boldsymbol{y} + (\boldsymbol{H} - \boldsymbol{I}_p)\boldsymbol{z}, \qquad \boldsymbol{z} \in \mathbb{R}^p$$
where $\boldsymbol{H} = (\boldsymbol{X}'\boldsymbol{X})^-(\boldsymbol{X}'\boldsymbol{X})$.

3.5 Properties of $\widehat{\beta}$

In the full-rank Gaussian linear model
$$\begin{aligned}E[\widehat{\boldsymbol{\beta}}] &= E[(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\mathcal{Y}]\\ &= (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'E[\mathcal{Y}]\\ &= (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\beta}\\ &= \boldsymbol{\beta}\end{aligned}$$

which is to say that the least squares estimator is an *unbiased estimator* of $\boldsymbol{\beta}$. Furthermore

$$\begin{aligned}\mathrm{Var}(\widehat{\boldsymbol{\beta}}) &= \mathrm{Var}\left((\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\mathcal{Y}\right)\\ &= (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\,\mathrm{Var}(\mathcal{Y})\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\\ &\sigma^2(\boldsymbol{X}'\boldsymbol{X})^{-1}\end{aligned}$$

R Exercises: Consider the models fit in Chap. 1

```
> lm1 <- lm(optden ~ 1 + carb, Formaldehyde)
> set.seed(1234)                       # allow for reproducible "random" numbers
> badDat <- within(data.frame(x1=1:20, x2=rnorm(20,mean=6,sd=0.2),
+                             x4=rexp(20,rate=0.02),
+                             y=runif(20,min=18,max=24)),
+                  x3 <- x1 + 2*x2)   # create linear combination
> lm2 <- lm(y ~ x1 + x2 + x3 + x4, badDat)
> lm3 <- lm(count ~ spray, InsectSprays)
> lmlst <- list(lm1=lm1, lm2=lm2, lm3=lm3)
> mmlst <- lapply(lmlst, model.matrix)
```

We know that models `lm1` and `lm3` are full-rank but model `lm2` is rank-deficient.

```
> sapply(lmlst, function(fm) c(rank=fm[["rank"]], p=length(coef(fm))))

     lm1 lm2 lm3
rank   2   4   6
p      2   5   6
```

which is reflected in the diagonal elements of the $\boldsymbol{R}$ matrices and in the singular values of the model matrices and in the condition number

```
> lapply(lmlst, function(fm) diag(fm[["qr"]][["qr"]]))

$lm1
[1] -2.4494897  0.6390097
$lm2
[1] -4.472136e+00  2.578759e+01 -8.668932e-01  2.327514e+02  5.179752e-15

$lm3
[1] -8.485281  3.162278  3.098387  3.000000  2.828427  2.449490

> lapply(mmlst, function(mm) svd(mm, nu=0, nv=0)[["d"]])

$lm1
[1] 2.773349 0.564389
$lm2
[1] 3.197628e+02 8.828492e+01 1.396147e+01 1.446151e-01 3.340320e-15

$lm3
[1] 9.069136 3.464102 3.464102 3.464102 3.464102 1.323169
```

```
> sapply(lmlst, kappa, exact=TRUE)

         lm1          lm2          lm3
4.913897e+00 1.512149e+17 6.854102e+00
```

For the full-rank models, `lm1` and `lm3`, the pseudo-inverse, $\boldsymbol{X}^-$ is simply the matrix the creates the estimated coefficients, $\widehat{\boldsymbol{\beta}}$ from the observed response vector $\boldsymbol{y}$. We can write it in various forms as

$$\boldsymbol{X}^- = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}' = \boldsymbol{R}^{-1}\boldsymbol{Q}_1' = \boldsymbol{V}\boldsymbol{D}^{-1}\boldsymbol{U}_1'$$

For full-rank models like these the pseudo-inverse, $\boldsymbol{X}^-$ is unique and

$$\boldsymbol{X}^-\boldsymbol{X} = \boldsymbol{R}^{-1}\underbrace{\boldsymbol{Q}_1'\boldsymbol{Q}_1}_{\boldsymbol{I}_p}\boldsymbol{R} = \boldsymbol{I}_p$$

We can verify the conditions for the Moore-Penrose inverse symbolically. For example,

$$\boldsymbol{X}\boldsymbol{X}^-\boldsymbol{X} = \boldsymbol{Q}_1\underbrace{\boldsymbol{R}\boldsymbol{R}^{-1}}_{\boldsymbol{I}_p}\underbrace{\boldsymbol{Q}_1'\boldsymbol{Q}_1}_{\boldsymbol{I}_p}\boldsymbol{R} = \boldsymbol{Q}_1\boldsymbol{R} = \boldsymbol{X}$$

or numerically

```
> X <- mmlst[[3]]
> lm3qr <- lmlst[[3]]$qr
> Q1 <- qr.Q(lm3qr)
> R <- qr.R(lm3qr)
> Xpinv <- backsolve(R, t(Q1))
> zapsmall(Xpinv %*% X)

     (Intercept) sprayB sprayC sprayD sprayE sprayF
[1,]           1      0      0      0      0      0
[2,]           0      1      0      0      0      0
[3,]           0      0      1      0      0      0
[4,]           0      0      0      1      0      0
[5,]           0      0      0      0      1      0
[6,]           0      0      0      0      0      1

> all.equal(X %*% Xpinv %*% X, X, check.attr=FALSE)

[1] TRUE

> all.equal(Xpinv %*% X %*% Xpinv, Xpinv, check.attr=FALSE)

[1] TRUE
```

For the rank-deficient model, `lm2`, there are many pseudo-inverses.

```
> X <- mmlst[[2]]
> lm2qr <- lmlst[[2]]$qr
> SVD <- svd(X)
> (rr <- lm2qr$rank)                        # rank

[1] 4

> (rrind <- seq_len(rr))                    # safer than 1:rr

[1] 1 2 3 4

> (dpinv <- c(1/SVD$d[rrind], rep(0, ncol(X) - rr)))

[1] 0.003127319 0.011326963 0.071625693 6.914905158 0.000000000

> str(Xpinv1 <- SVD$v %*% (dpinv * t(SVD$u)))

 num [1:5, 1:20] 1.245927 0.048805 -0.055057 -0.061309 -0.000183 ...

> zapsmall(Xpinv1 %*% X)

     (Intercept)         x1         x2        x3 x4
[1,]           1  0.0000000  0.0000000 0.0000000  0
[2,]           0  0.8333333 -0.3333333 0.1666667  0
[3,]           0 -0.3333333  0.3333333 0.3333333  0
[4,]           0  0.1666667  0.3333333 0.8333333  0
[5,]           0  0.0000000  0.0000000 0.0000000  1

> all.equal(X %*% Xpinv1 %*% X, X, check.attr=FALSE)

[1] TRUE

> all.equal(Xpinv1 %*% X %*% Xpinv1, Xpinv1, check.attr=FALSE)

[1] TRUE

> ## An alternative construction is to reduce the SVD components to the first 4 columns
> str(SVDred <- list(d=SVD$d[rrind], u=SVD$u[,rrind], v=SVD$v[,rrind]))

List of 3
 $ d: num [1:4] 319.763 88.285 13.961 0.145
 $ u: num [1:20, 1:4] 0.0262 0.0559 0.155 0.0498 0.1678 ...
 $ v: num [1:5, 1:4] 0.00901 0.11732 0.05349 0.2243 0.96591 ...

> str(Xpinv2 <- with(SVDred, v %*% (1/d * t(u))))

 num [1:5, 1:20] 1.245927 0.048805 -0.055057 -0.061309 -0.000183 ...

> all.equal(Xpinv2, Xpinv1)
```

```
[1] TRUE

> ## Finally, we can use a similar construction on the QR decomposition
> ## taking into account the rearrangement of the columns of X
> Xpiv <- X[, lm2qr$pivot]
> str(Xpinv3 <- rbind(backsolve(qr.R(lm2qr)[rrind, rrind], t(qr.Q(lm2qr)[, rrind])), 0))

 num [1:5, 1:20] 1.245927 -0.012504 -0.177675 -0.000183 0 ...

> all.equal(Xpiv %*% Xpinv3 %*% Xpiv, Xpiv, check.attr=FALSE)

[1] TRUE

> all.equal(Xpinv3 %*% Xpiv %*% Xpinv3, Xpinv3, check.attr=FALSE)

[1] TRUE
```

The last two constructions show that the Moore-Penrose pseudo-inverse is a matter of collecting the independent columns at the left hand side of the matrix and the linearly-dependent columns on the right hand side, then truncating the decomposition. In other words, is $\boldsymbol{X}$ is less than full rank then you just find a set of full-rank columns and proceed as before.

R Exercise: (Simulating linear model fits) The `simulate` functions allow us to simulate a matrix of responses based on a fitted model, then fit all the simulated responses in a single call to `lm`. This is much, much faster than any loop-based approach would be.

The result of `simulate` is a named list of response vectors so we drop the names and convert the list to a matrix.

```
> str(Ymat <- data.matrix(unname(simulate(lm1, 10000))))

 num [1:6, 1:10000] 0.0843 0.2584 0.4324 0.5263 0.6142 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:6] "1" "2" "3" "4" ...
  ..$ : NULL

> fits <- lm(Ymat ~ carb, Formaldehyde)
> str(coefs <- coef(fits))

 num [1:2, 1:10000] -0.00201 0.87284 -0.00922 0.90215 0.00369 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:2] "(Intercept)" "carb"
  ..$ : NULL
```

Most of the time we want the coefficients to be a data frame instead with columns corresponding to the coefficient names.

```
> str(coefs <- data.frame(t(coef(fits)), check.names=FALSE))
```

```
'data.frame':        10000 obs. of  2 variables:
 $ (Intercept): num  -0.002005 -0.009224 0.003691 -0.001113 0.000131 ...
 $ carb       : num  0.873 0.902 0.887 0.896 0.892 ...
```

Recall that the "true" coefficients for this model are

```
> printCoefmat(coef(summary(lm1)))

             Estimate Std. Error t value  Pr(>|t|)
(Intercept) 0.0050857  0.0078337  0.6492    0.5516
carb        0.8762857  0.0135345 64.7444 3.409e-07
```

For an unbiased estimator the mean of the distribution of the estimator should be the parameter value.

```
> sapply(coefs, mean)

(Intercept)        carb
0.005078325 0.876313251
```

and the standard deviations should be close to the standard errors

```
> sapply(coefs, sd)

(Intercept)        carb
0.007943042 0.013647517
```

The correlation of sample of coefficient estimates should be close to the value for the fitted model

```
> summary(lm1, corr=TRUE)$correlation

            (Intercept)      carb
(Intercept)    1.000000 -0.892664
carb          -0.892664  1.000000

> cor(coefs)

            (Intercept)      carb
(Intercept)    1.000000 -0.896498
carb          -0.896498  1.000000
```

If, instead, we wish to consider the variance-covariance matrices, we use

```
> vcov(lm1)

              (Intercept)          carb
(Intercept)  6.136653e-05 -0.0000946449
carb        -9.464490e-05  0.0001831837

> var(coefs)
```

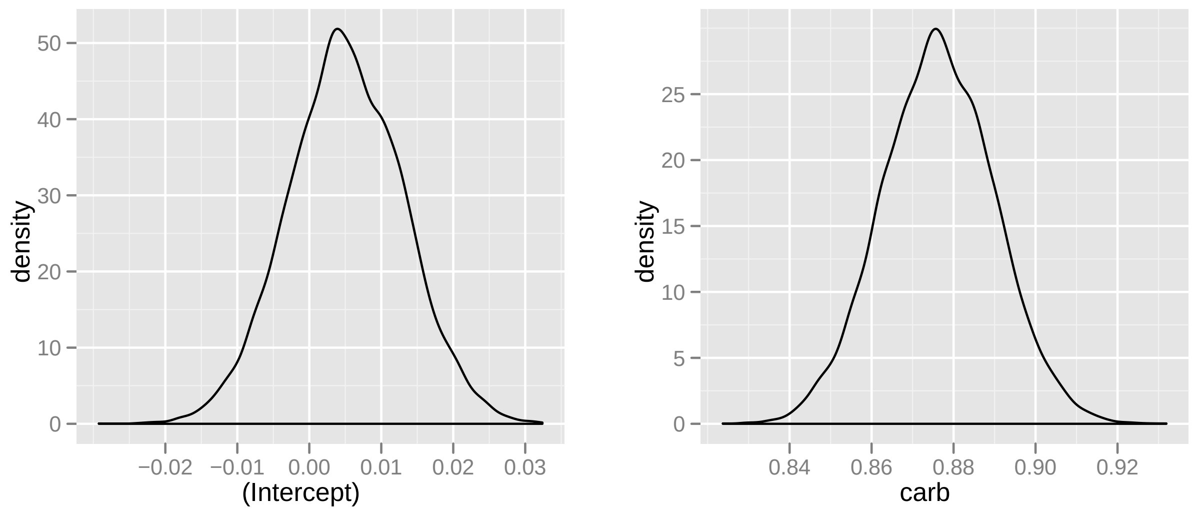

Figure 3.1: Empirical density plots of coefficient estimates from data simulated according to the estimated parameters in model `lm1`

```
              (Intercept)          carb
(Intercept)  6.309192e-05 -0.0000971829
carb        -9.718290e-05  0.0001862547
```

In Fig. 3.1 we show the empirical density plots for the coefficients separately Alternatively, we could examine the normal Q-Q plots (Fig. 3.2).

We could also plot contours of the estimated 2-dimensional density (Fig. 3.3) The background of the empirical density contours is like a two-dimensional histogram but using hexagonal shaped bins instead of rectangles.

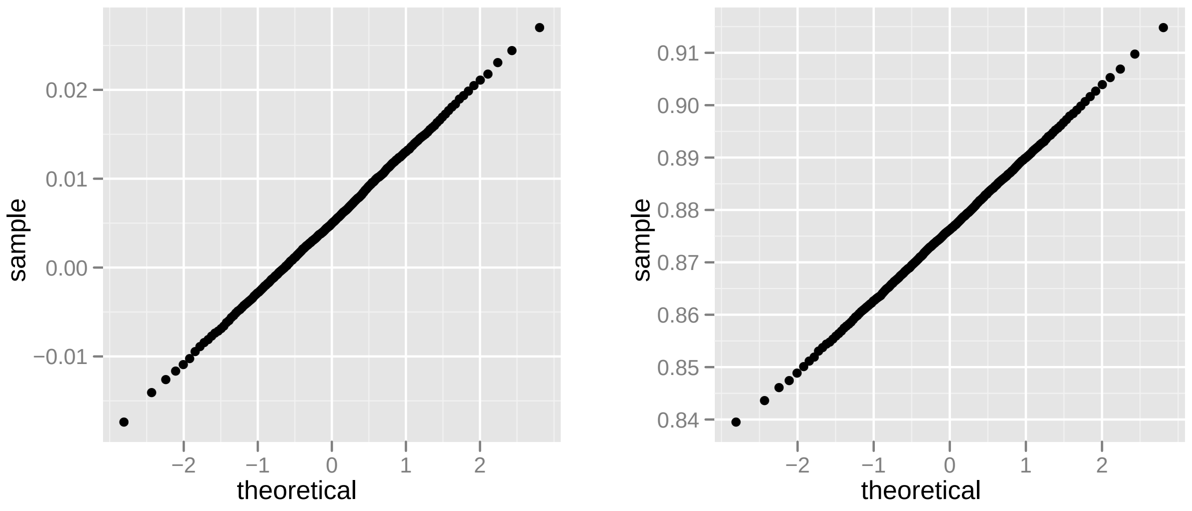

Figure 3.2: Normal quantile-quantile plots of coefficient estimates from responses simulated according to the estimated parameters in model `lm1`.

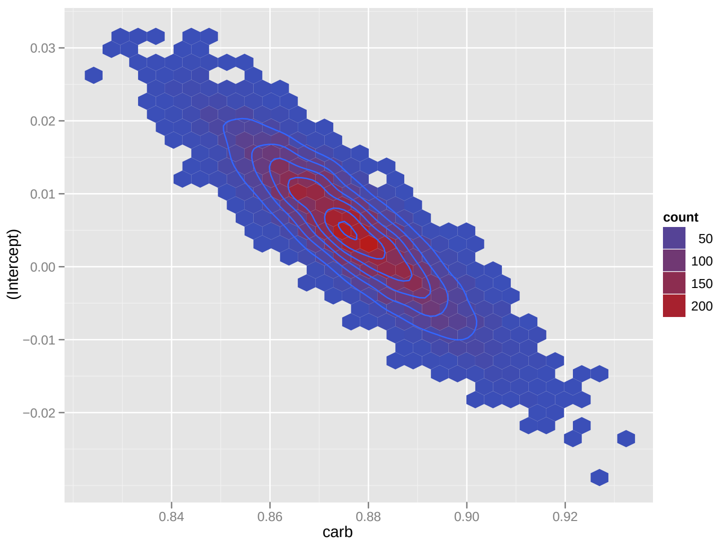

Figure 3.3: Normal quantile-quantile plots of coefficient estimates from responses simulated according to the estimated parameters in model `lm1`.