Statistics 849 notes – Fall 2010

Douglas Bates

Based on notes by Sündüz Keleş from Fall 2009
that were transcribed by Brittany Schwefel

September 17, 2010

Contents

1 The Gaussian Linear Model 1

1.1 Gaussian Linear Model 1

1.2 Linear algebra of least squares 2

1.3 Matrix decompositions 3

1.3.1 Orthogonal matrices 3

1.3.2 The QR decomposition 3

1.3.3 Comparison to the usual text-book formulas 12

1.3.4 R functions related to the QR decomposition 15

1.4 Related matrix decompositions 16

1.4.1 The Cholesky decomposition 16

1.4.2 Evaluation of the Cholesky decomposition 16

1.4.3 The singular value decomposition 17

1.5 Theoretical results on the eigendecomposition 21

1.5.1 Eigenvalues and Eigenvectors 21

1.5.2 Diagonalization of a Symmetric Matrix 22

1.5.3 Spectral Decomposition 23

1.5.4 Trace and Determinant of A 23

2 Quadratic Forms 25

2.1 Quadratic Forms 25

2.2 Idempotent Matrices 26

2.3 Expected Values and Covariance Matrices of Random Vectors 27

2.4 Mean and Variance of Quadratic Forms 28

2.5 Distribution of Quadratic Forms in Normal Random Variables 28

3 Properties of coefficient estimates 35

3.1 Geometric Properties 35

3.2 Calculus Approach 36

3.3 Algebraic Properties of $\widehat{\beta}$ 36

3.4 Rank deficient cases and the Moore-Penrose inverse 37

3.4.1 Properties of Generalized Inverses 37

3.5 Properties of $\widehat{\beta}$ 38

4 The Gauss-Markov Theorem 47

Chapter 1

The Gaussian Linear Model

Statistics 849 is part of a two-semester sequence, 849 & 850, on *Theory and Applications of Regression and Analysis of Variance*. In these courses we study statistical models that relate a *response* to values of one or more *covariates*, which are variables that are observed in conjunction with the response.

The statistical inferences are based on a probability model that characterizes the distribution of the vector-valued *random variable*, $\mathcal{Y}$, as it depends on values of the covariates. We build the model based on observed values of the responses, represented by the vector $\boldsymbol{y}$, and corresponding values of the covariates.

All the models we will study are based on a *linear predictor* expression, $\boldsymbol{X\beta}$, where the $n \times p$ matrix $\boldsymbol{X}$ is the *model matrix* created from a model specification and the values of the covariates. Here n is the number of observations and p is the dimension of the *coefficient vector*, $\boldsymbol{\beta}$, The coefficients are *parameters* in the model. We form *estimates*, $\widehat{\boldsymbol{\beta}}$, of these parameters from the observed data.

We assume that $n \geq p$. That is, we have at least as many observations are we have coefficients in the model.

1.1 Gaussian Linear Model

A basic model for a response, $\mathcal{Y}$, that is measured on a continuous scale, is the *Gaussian Linear Model*

$$\mathcal{Y} \sim \mathcal{N}(\boldsymbol{X\beta}_T, \sigma^2\boldsymbol{I}_n) \tag{1.1}$$

where $\boldsymbol{I}_n$ is the n-dimensional identity matrix, $\boldsymbol{\beta}_T$ is the "true", but unknown, value of the coefficient vector and $\mathcal{N}$ denotes the multivariate Gaussian (also called *normal*) distribution.

The probability density of $\mathcal{Y}$,

$$f_{\mathcal{Y}}(\boldsymbol{y}) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(\frac{-\|\boldsymbol{y} - \boldsymbol{X\beta}_T\|^2}{2\sigma^2}\right), \tag{1.2}$$

is called a *spherical normal* density, because contours of constant density are concentric spheres centered at $\boldsymbol{X\beta}_T$.

The *likelihood*, $L(\boldsymbol{\beta}, \sigma|\boldsymbol{y})$, of the parameters, $\boldsymbol{\beta}$ and σ, given the observed responses, $\boldsymbol{y}$, and the model matrix, $\boldsymbol{X}$, is the same expression as the probability density, $f_{\mathcal{Y}}(\boldsymbol{y})$, but regarded as a function of the parameters given the data, as opposed to the density, which is a function of $\boldsymbol{y}$ for known values of the parameters.

$$L(\boldsymbol{\beta}, \sigma|\boldsymbol{y}) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(\frac{-\|\boldsymbol{y} - \boldsymbol{X\beta}\|^2}{2\sigma^2}\right). \tag{1.3}$$

The *maximum likelihood estimates* (*mle*s) of the parameters are, as the name suggests, the values of the parameters that maximize the likelihood

$$\left(\widehat{\boldsymbol{\beta}}', \widehat{\sigma}_L\right)' = \arg\max_{\boldsymbol{\beta},\sigma} L(\boldsymbol{\beta}, \sigma|\boldsymbol{y}) \tag{1.4}$$

As often happens it is much easier to maximize the expression for the *log-likelihood*

$$\ell(\boldsymbol{\beta}, \sigma|\boldsymbol{y}) = \log\left(L(\boldsymbol{\beta}, \sigma|\boldsymbol{y})\right) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{\|\boldsymbol{y} - \boldsymbol{X\beta}\|^2}{2\sigma^2} \tag{1.5}$$

than to maximize the likelihood. Because the logarithm function is monotonic increasing, the values of the parameters that maximize the log-likelihood, written $\arg\max_{\beta,\sigma} \ell(\boldsymbol{\beta}, \sigma|\boldsymbol{y})$, are exactly the same as the values that maximize the likelihood, $\arg\max_{\beta,\sigma} L(\boldsymbol{\beta}, \sigma|\boldsymbol{y})$.

The expression can be simplified further by converting to the *deviance*, which is negative twice the log-likelihood,

$$d(\boldsymbol{\beta}, \sigma|\boldsymbol{y}) = -2\ell(\boldsymbol{\beta}, \sigma|\boldsymbol{y}) = n\log(2\pi\sigma^2) + \frac{\|\boldsymbol{y} - \boldsymbol{X\beta}\|^2}{\sigma^2}. \tag{1.6}$$

Because of the negative sign, the mle's are the values that minimize the deviance. For any fixed value of σ^2, the deviance is minimized with respect to $\boldsymbol{\beta}$ when the *residual sum of squares*,

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

is minimized. Thus the mle of the coefficient vector, $\widehat{\boldsymbol{\beta}}$, in the Gaussian linear model is the *least squares estimate*

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X\beta}\|^2. \tag{1.7}$$

1.2 Linear algebra of least squares

Because the Gaussian Linear Model, $\mathcal{Y} \sim \mathcal{N}(\boldsymbol{X\beta}_T, \sigma^2\boldsymbol{I}_n)$, is intimately tied to the Euclidean distance, $\|\boldsymbol{y} - \boldsymbol{X\beta}\|^2$, and because the set of all possible fitted values, $\{\boldsymbol{X\beta} : \boldsymbol{\beta} \in \mathbb{R}^p\}$, which is called the *column span* of $\boldsymbol{X}$ and written $\text{col}(\boldsymbol{X})$, is a linear subspace of $\mathbb{R}^n$, linear algebra, especially as related to the model matrix, $\boldsymbol{X}$, and the response vector, $\boldsymbol{y}$, is fundamental to the theory and practice of linear regression analysis.

We will concentrate on the theoretical and computational aspects of linear algebra as related to the linear model and the implementation of such models in R.

1.3 Matrix decompositions

1.3.1 Orthogonal matrices

An *orthogonal* $n \times n$ matrix, $\boldsymbol{Q}$ has the property that its transpose is its inverse,

$$\boldsymbol{Q}'\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{Q}' = \boldsymbol{I}_n.$$

These properties imply that the columns of $\boldsymbol{Q}$ must be orthogonal to each other and must all have unit length. The same is true for the rows.

An orthogonal matrix has a special property that it preserves lengths.

Preserving lengths

For any $\boldsymbol{x} \in \mathbb{R}^n$

$$\|\boldsymbol{Q}\boldsymbol{x}\|^2 = (\boldsymbol{Q}\boldsymbol{x})'\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{x}'\boldsymbol{Q}'\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{x}'\boldsymbol{x} = \|\boldsymbol{x}\|^2$$

Thus the linear transformation determined by $\boldsymbol{Q}$ or by $\boldsymbol{Q}'$ must be a *rigid* transformation, composed of reflections or rotations.

Orthogonal transformations of the response space, $\mathbb{R}^n$, will be important to us because they preserve lengths and because the likelihood of the parameters, $\boldsymbol{\beta}$, is related to the squared length of the residual vector, $\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2$.

1.3.2 The QR decomposition

Any $n \times p$ matrix $\boldsymbol{X}$ has a QR decomposition consisting of an orthogonal $n \times n$ matrix $\boldsymbol{Q}$ and a $p \times p$ matrix $\boldsymbol{R}$ that is zero below the main diagonal (in other words, it is *upper triangular*). The QR decomposition of the model matrix $\boldsymbol{X}$ is written

$$\boldsymbol{X} = \boldsymbol{Q}\begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{bmatrix} = \begin{bmatrix}\boldsymbol{Q}_1 & \boldsymbol{Q}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{bmatrix} = \boldsymbol{Q}_1\boldsymbol{R} \tag{1.8}$$

where $\boldsymbol{Q}_1$ is the first p columns of $\boldsymbol{Q}$ and $\boldsymbol{Q}_2$ is the last $n - p$ columns of $\boldsymbol{Q}$.

That fact that matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ must exist is proved by construction. The matrix $\boldsymbol{Q}$ is the product of p *Householder reflections* (see the Wikipedia page for the QR decomposition). The process of generating the upper triangular matrix $\boldsymbol{R}$ is similar to the Gram-Schmidt orthogonalization process, but more flexible and more numerically stable. If the diagonal elements of $\boldsymbol{R}$ are all non-zero (in practice this means that none of them are very small in absolute value) then $\boldsymbol{X}$ has *full column rank* and the columns of $\boldsymbol{Q}_1$ form an *orthonormal basis* for $\mathrm{col}(\boldsymbol{X})$.

The implementation of the QR decomposition in R guarantees that any elements on the diagonal of $\boldsymbol{R}$ that are considered effectively zero are rearranged by column permutation to occur in the trailing columns. That is, if the rank of $\boldsymbol{X}$ is $k < p$ then the first k columns of $\boldsymbol{Q}$ form an orthonormal basis for $\mathrm{col}(\boldsymbol{X}\boldsymbol{P})$ where $\boldsymbol{P}$ is a $p \times p$ permutation matrix, which means that it is a rearrangement of the columns of $\boldsymbol{I}_p$.

Our text often mentions rank-deficient cases where $\operatorname{rank}(\boldsymbol{X}) = k < p$. In practice the rank deficient case rarely occurs because the process of building the model matrix in R involves a considerable amount of analysis of the model formula to remove the most common cases of rank deficiency. Nevertheless, rank deficiency can occur and is detected and handled in the `lm` function in R.

Because multiplication by an orthogonal matrix like $\boldsymbol{Q}'$ preserves lengths we can write

$$\begin{aligned}\widehat{\boldsymbol{\beta}} &= \arg\min_{\beta} \|\boldsymbol{y} - \boldsymbol{X\beta}\|^2 \\ &= \arg\min_{\beta} \|\boldsymbol{Q}'(\boldsymbol{y} - \boldsymbol{X\beta})\|^2 \\ &= \arg\min_{\beta} \|\boldsymbol{Q}'\boldsymbol{y} - \boldsymbol{Q}'\boldsymbol{X\beta}\|^2 \\ &= \arg\min_{\beta} \|\boldsymbol{c}_1 - \boldsymbol{R\beta}\|^2 + \|\boldsymbol{c}_2\|^2\end{aligned} \tag{1.9}$$

where $\boldsymbol{c}_1 = \boldsymbol{Q}_1'\boldsymbol{y}$ is the first p elements of $\boldsymbol{Q}'\boldsymbol{y}$ and $\boldsymbol{c}_2 = \boldsymbol{Q}_2'\boldsymbol{y}$ is the last $n - p$ elements. If $\operatorname{rank}(\boldsymbol{X}) = p$ then $\operatorname{rank}(\boldsymbol{R}) = p$ and $\boldsymbol{R}^{-1}$ exists so we can write $\widehat{\boldsymbol{\beta}} = \boldsymbol{R}^{-1}\boldsymbol{c}_1$ (although you don't actually calculate $\boldsymbol{R}^{-1}$ to solve the triangular linear system $\boldsymbol{R}\widehat{\boldsymbol{\beta}} = \boldsymbol{c}_1$ for $\widehat{\boldsymbol{\beta}}$).

In a model fit by the `lm()` or `aov()` functions in R there is a component `$effects` which is $\boldsymbol{Q}'\boldsymbol{y}$. The component `$qr` is a condensed form of the QR decomposition of the model matrix $\boldsymbol{X}$. The matrix $\boldsymbol{R}$ is embedded in there but the matrix $\boldsymbol{Q}$ is a virtual matrix represented as a product of Householder reflections and not usually evaluated explicitly.

R Exercise: To see this theory in action, we will start with a very simple linear model. The `Formaldehyde` data are six observations from a calibration experiment. The response, `optden`, is the optical density. Only one covariate, `carb`, which is the carbohydrate concentration, is included in the data frame.

```
> str(Formaldehyde)

'data.frame':        6 obs. of  2 variables:
 $ carb  : num  0.1 0.3 0.5 0.6 0.7 0.9
 $ optden: num  0.086 0.269 0.446 0.538 0.626 0.782
```

The `model.matrix()` function extracts the model matrix, $\boldsymbol{X}$, from a fitted linear model object

```
> (X <- model.matrix(lm1 <- lm(optden ~ 1 + carb, Formaldehyde)))

  (Intercept) carb
1           1  0.1
2           1  0.3
3           1  0.5
4           1  0.6
5           1  0.7
6           1  0.9
attr(,"assign")
[1] 0 1
```

The `$qr` component is an object of class `"qr"`

```
> class(qrlm1 <- lm1$qr)
```

```
[1] "qr"
```

for which there are many extractor functions and methods (see `?qr`).

```
> (R <- qr.R(qrlm1))
```

```
  (Intercept)       carb
1   -2.449490 -1.2655697
2    0.000000  0.6390097
```

produces the $\boldsymbol{R}$ matrix while

```
> (Q1 <- qr.Q(qrlm1))
```

```
           [,1]        [,2]
[1,] -0.4082483 -0.65205066
[2,] -0.4082483 -0.33906635
[3,] -0.4082483 -0.02608203
[4,] -0.4082483  0.13041013
[5,] -0.4082483  0.28690229
[6,] -0.4082483  0.59988661
```

by default produces $\boldsymbol{Q}_1$. First we should check that their product is indeed $\boldsymbol{X}$

```
> (Q1R <- Q1 %*% R)
```

```
     (Intercept) carb
[1,]           1  0.1
[2,]           1  0.3
[3,]           1  0.5
[4,]           1  0.6
[5,]           1  0.7
[6,]           1  0.9
```

It seems to be the same, although as in all floating point calculations on a computer, there may be some small imprecision caused by round-off error in the calculations. This is why we don't use exact comparisons on the results of floating point calculations

```
> all(X == Q1R)
```

```
[1] FALSE
```

but instead compare results using

```
> all.equal(X, Q1R, check.attr = FALSE)
```

```
[1] TRUE
```

(As a model matrix, `X` has some additional attributes that are not present in the product `Q1R`, which is why we turn off checking of the attributes of the two objects.)

Notice that $\boldsymbol{X}$ and $\boldsymbol{y}$ are not explicitly part of the fitted model object, `lm1`.

```
> names(lm1)

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

Both are generated from the *model.frame*, which is stored as the component `$model`. Althought this is getting into more detail than is needed at present, the reason for introducing the model frame is to say that the safe way of extracting the response vector, $\boldsymbol{y}$, is

```
> (y <- model.response(model.frame(lm1)))

    1     2     3     4     5     6
0.086 0.269 0.446 0.538 0.626 0.782
```

We have already seen that `model.matrix` returns the matrix $\boldsymbol{X}$ from the fitted model object.

We can produce the full $n \times n$ orthogonal matrix $\boldsymbol{Q}$ from `qr.Q()` by setting the optional argument `complete=TRUE`. We do this for illustration only. In practice the matrix $\boldsymbol{Q}$ is never explicitly created — it is a "virtual" matrix in the sense that it is a product of Householder reflections that are stored much more compactly than $\boldsymbol{Q}$ would be stored.

```
> (Q <- qr.Q(qrlm1, complete=TRUE))

           [,1]        [,2]        [,3]       [,4]       [,5]       [,6]
[1,] -0.4082483 -0.65205066 -0.37370452 -0.3405290 -0.3073534 -0.2410023
[2,] -0.4082483 -0.33906635  0.05460995  0.2207196  0.3868293  0.7190487
[3,] -0.4082483 -0.02608203  0.86857638 -0.1439791 -0.1565346 -0.1816455
[4,] -0.4082483  0.13041013 -0.15359661  0.8125532 -0.2212971 -0.2889976
[5,] -0.4082483  0.28690229 -0.17576960 -0.2309146  0.7139404 -0.3963496
[6,] -0.4082483  0.59988661 -0.22011559 -0.3178501 -0.4155847  0.3889463
```

The `$effects` vector should be the product $\boldsymbol{Q}'\boldsymbol{y}$ but it happens that they are stored differently. The `$effects` vector is a vector of length n and the product $\boldsymbol{Q}'\boldsymbol{y}$ is an $n \times 1$ matrix. To compare them, we need to make `$effects` an $n \times 1$ matrix or make $\boldsymbol{Q}'\boldsymbol{y}$ into a vector. A convenient way of making an $n \times 1$ matrix from an n-vector is the function `cbind()`, which creates matrices or data frames by binding columns together. If we give it a single vector argument it creates an $n \times 1$ matrix

```
> str(cbind(lm1$effects))
```

```
 num [1:6, 1] -1.12146 0.55996 0.00514 0.00992 0.01069 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:6] "(Intercept)" "carb" "" "" ...
  ..$ : NULL
```

```
> str(crossprod(Q, y))
```

```
 num [1:6, 1] -1.12146 0.55996 0.00514 0.00992 0.01069 ...
```

```
> all.equal(cbind(lm1$effects), crossprod(Q, y), check.attr=FALSE)
```

```
[1] TRUE
```

(The function `crossprod(A,B)` creates $\boldsymbol{A}'\boldsymbol{B}$ directly, without creating $\boldsymbol{A}'$ from $\boldsymbol{A}$. It is most commonly used to create matrices like $\boldsymbol{X}'\boldsymbol{X}$ as

```
> crossprod(X)
```

```
            (Intercept) carb
(Intercept)         6.0 3.10
carb                3.1 2.01
```

The companion function, `tcrossprod`, creates $\boldsymbol{X}\boldsymbol{X}'$.)

If we wish to do the comparison by converting $\boldsymbol{Q}'\boldsymbol{y}$ to a vector, we can use

```
> all.equal(lm1$effects, as.vector(crossprod(Q, y)), check.attr=FALSE)
```

```
[1] TRUE
```

I find `cbind` easier to type than `as.vector`.

Another way of generating $\boldsymbol{Q}'\boldsymbol{y}$ is with the function `qr.qty()`

```
> all.equal(lm1$effects, qr.qty(qrlm1, y), check.attr=FALSE)
```

```
[1] TRUE
```

We should check that $\boldsymbol{Q}$ is indeed orthogonal and that $\boldsymbol{Q}_1'\boldsymbol{Q}_1 = \boldsymbol{I}_p$. The matrix $\boldsymbol{I}_k$ is generated by `diag(nrow=k)`.

```
> all.equal(crossprod(Q1), diag(nrow=ncol(Q1)))
```

```
[1] TRUE
```

```
> all.equal(crossprod(Q), diag(nrow=nrow(Q)))
```

```
[1] TRUE
```

```
> all.equal(tcrossprod(Q), diag(nrow=nrow(Q)))
```

```
[1] TRUE
```

When we print a matrix that may have negligibly small non-zero values in it

```
> crossprod(Q1)

             [,1]         [,2]
[1,] 1.000000e+00 1.197637e-16
[2,] 1.197637e-16 1.000000e+00
```

we can clean up the output with `zapsmall()` which, as the name suggests, zeros the very small values.

```
> zapsmall(crossprod(Q1))

     [,1] [,2]
[1,]    1    0
[2,]    0    1

> zapsmall(crossprod(Q))

     [,1] [,2] [,3] [,4] [,5] [,6]
[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
```

Because the diagonal elements of $\boldsymbol{R}$ are all safely non-zero, we can solve the system $\boldsymbol{R}\widehat{\boldsymbol{\beta}} = \boldsymbol{Q}_1'\boldsymbol{y}$ for the coefficient estimates, $\widehat{\boldsymbol{\beta}}$. We could use the function `solve()` to do this but it us better to use `backsolve()` for the solution to upper triangular systems,

```
> coef(lm1)

(Intercept)        carb
0.005085714 0.876285714

> backsolve(R, crossprod(Q1, y))

            [,1]
[1,] 0.005085714
[2,] 0.876285714

> all.equal(coef(lm1), as.vector(backsolve(R, crossprod(Q1, y))), check.attr=FALSE)

[1] TRUE
```

The function `qr.coef` combines the multiplication of $\boldsymbol{y}$ by $\boldsymbol{Q}_1'$ and the backsolve step

```
> qr.coef(qrlm1, y)

(Intercept)        carb
0.005085714 0.876285714
```

R Exercise: As seen above, a linear model is specified as a model formula and the data frame in which to evaluate the formula. Because the formula is analyzed for conditions that may introduce rank deficiency and consequently removes those conditions, rank deficient cases occur infrequently. Of course, it is possible to artificially generate data with a built-in rank dependency

```
> 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
> (summary(lm2 <- lm(y ~ x1 + x2 + x3 + x4, badDat)))

Call:
lm(formula = y ~ x1 + x2 + x3 + x4, data = badDat)
Residuals:
    Min      1Q  Median      3Q     Max
-2.3444 -1.7670 -0.3585  1.6159  3.0292

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.793e+01  1.390e+01   1.290    0.215
x1          4.140e-02  8.553e-02   0.484    0.635
x2          3.822e-01  2.358e+00   0.162    0.873
x3                 NA         NA      NA       NA
x4          3.901e-05  8.680e-03   0.004    0.996

Residual standard error: 2.02 on 16 degrees of freedom
Multiple R-squared: 0.021,        Adjusted R-squared: -0.1626
F-statistic: 0.1144 on 3 and 16 DF,  p-value: 0.9504

> (lm2qr <- lm2$qr)$rank

[1] 4

> qr.R(lm2qr)                         # the columns are rearranged

  (Intercept)        x1          x2         x4            x3
1   -4.472136 -46.95743 -26.6086150 -182.87421 -1.001747e+02
2    0.000000  25.78759   0.1721295   83.85993  2.613185e+01
3    0.000000   0.00000  -0.8668932   35.62409 -1.733786e+00
4    0.000000   0.00000   0.0000000  232.75139  2.005660e-15
5    0.000000   0.00000   0.0000000    0.00000  5.179752e-15

> lm2qr$pivot                         # the permutation vector

[1] 1 2 3 5 4
```

but we don't do this in practice.

The common case of an analysis of variance model which, when written as

$$y_{i,j} = \mu + \alpha_i + \epsilon_{i,j} \quad i = 1, \dots, I,\ j = 1, \dots, n_i$$

would generate linearly dependent columns for μ and the α_i, $i = 1, \dots, I$ is analyzed and represented by the intercept column and $I - 1$ columns for the factor.

```
> str(InsectSprays)

'data.frame':        72 obs. of  2 variables:
 $ count: num  10 7 20 14 14 12 10 23 17 20 ...
 $ spray: Factor w/ 6 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...

> unique(mm <- model.matrix(lm3 <- lm(count ~ spray, InsectSprays)))

   (Intercept) sprayB sprayC sprayD sprayE sprayF
1            1      0      0      0      0      0
13           1      1      0      0      0      0
25           1      0      1      0      0      0
37           1      0      0      1      0      0
49           1      0      0      0      1      0
61           1      0      0      0      0      1

> attr(mm, "assign")

[1] 0 1 1 1 1 1

> qr.R(lm3[["qr"]])

  (Intercept)    sprayB     sprayC     sprayD     sprayE     sprayF
1   -8.485281 -1.414214 -1.4142136 -1.4142136 -1.4142136 -1.4142136
2    0.000000  3.162278 -0.6324555 -0.6324555 -0.6324555 -0.6324555
3    0.000000  0.000000  3.0983867 -0.7745967 -0.7745967 -0.7745967
4    0.000000  0.000000  0.0000000  3.0000000 -1.0000000 -1.0000000
5    0.000000  0.000000  0.0000000  0.0000000  2.8284271 -1.4142136
6    0.000000  0.000000  0.0000000  0.0000000  0.0000000  2.4494897
```

This shows only the unique rows in the model matrix. The six levels of the `spray` factor are represented by 5 indicator columns.

Because we are discussing an analysis of variance model we also show the `"assign"` attribute of the model matrix. This indicates that the first column is associated with the 0th term, which is the intercept, and the second through sixth columns are associated with the first term, which is `spray`.

In general, a factor with I levels is converted to a set of $I-1$ columns. These are called *contrasts*, but be warned that these do not fulfill the definition of contrasts as used in some texts. You should think of them as being a set of columns representing changes between levels of the factor.

The type of contrasts generated is controlled by the option called `"contrasts"`.

```
> getOption("contrasts")

        unordered           ordered
"contr.treatment"      "contr.poly"
```

(By the way, plotting these data first would show that this is not a good model. The `count` variable is, not surprisingly, a count and does not have constant variance. A better model would use the square root of the count as a response.)

The determinant of an orthogonal matrix

As described on its Wikipedia page, the *determinant*, $|\boldsymbol{A}|$, of the $k \times k$ square matrix, $\boldsymbol{A}$, is the volume of the parallelepiped spanned by its columns (or, equivalently, the volume spanned by its rows). Because we can consider either the rows or the columns when evaluating the determinant, we must have

$$|\boldsymbol{A}| = |\boldsymbol{A}'|.$$

We can regard $|\boldsymbol{A}|$ as the magnification factor in the transformation $\boldsymbol{x} \rightarrow \boldsymbol{A}\boldsymbol{x}$ from $\mathbb{R}^k$ to $\mathbb{R}^k$. This transformation takes the unit cube to a parallelopiped with volume $|\boldsymbol{A}|$. Composing transformations will just multiply the magnification factors so we must have

$$|\boldsymbol{A}\boldsymbol{B}| = |\boldsymbol{A}|\,|\boldsymbol{B}|$$

We know that the columns of an orthogonal matrix $\boldsymbol{Q}$ are *orthonormal* hence they span a unit volume. That is, for an $n \times n$ matrix $\boldsymbol{Q}$

$$\boldsymbol{Q}'\boldsymbol{Q} = \boldsymbol{I}_n \quad \Rightarrow \quad |\boldsymbol{Q}| = \pm 1.$$

The sign indicates whether the transformation preserves orientation. In two dimensions a rotation preserves orientation and a reflection reverses orientation.

Furthermore, the determinant of a *diagonal* matrix or a *triangular* matrix is simply the product of its diagonal elements. (For a triangular matrix, first consider the 2×2 case and the shape of the parallelogram spanned by the columns. The width of the parallelogram is the (1,1) element and the height is the (2,2) element so the area is the product of the diagonal elements (up to sign). Then convince yourself that this property scales to a parellelopiped in k dimensions.)

From these properties we can formally derive

$$1 = |\boldsymbol{I}_n| = |\boldsymbol{Q}\boldsymbol{Q}'| = |\boldsymbol{Q}||\boldsymbol{Q}'| = |\boldsymbol{Q}|^2 \quad \Rightarrow \quad |\boldsymbol{Q}| = \pm 1.$$

Interestingly, one way that the determinant, $|\boldsymbol{A}|$ is evaluated in practice is by forming the QR decomposition of $\boldsymbol{A}$, taking the product of the diagonal elements of $\boldsymbol{R}$, and determining whether $|\boldsymbol{Q}|$ has a plus or a minus sign.

R Exercise: The `det()` function evaluates the determinant of a square matrix (although if you check its definition you will find that it just calls another function `determinant`, which is the preferred approach).

```
> all.equal(det(R), prod(diag(R)))

[1] TRUE

> det(crossprod(Q1))

[1] 1

> det(crossprod(Q))

[1] 1
```

1.3.3 Comparison to the usual text-book formulas

Most text books state that the least squares estimates are

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y} \tag{1.10}$$

giving the impression that $\widehat{\boldsymbol{\beta}}$ is calculated this way. It isn't.

If you substitute $\boldsymbol{X} = \boldsymbol{Q}_1\boldsymbol{R}$ in eqn. 1.10 you get

$$(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y} = (\boldsymbol{R}'\boldsymbol{R})^{-1}\boldsymbol{R}'\boldsymbol{Q}_1'\boldsymbol{y} = \boldsymbol{R}^{-1}(\boldsymbol{R}')^{-1}\boldsymbol{R}'\boldsymbol{Q}_1'\boldsymbol{y} = \boldsymbol{R}^{-1}\boldsymbol{Q}_1'\boldsymbol{y},$$

our previous result.

Whenever you see $\boldsymbol{X}'\boldsymbol{X}$ in a formula you should mentally replace it by $\boldsymbol{R}'\boldsymbol{R}$ and similarly replace $(\boldsymbol{X}'\boldsymbol{X})^{-1}$ by $\boldsymbol{R}^{-1}(\boldsymbol{R}')^{-1}$ then see if you can simplify the result.

For example, the variance of the least squares estimator $\widehat{\boldsymbol{\beta}}$ is

$$\mathrm{Var}(\widehat{\boldsymbol{\beta}}) = \sigma^2(\boldsymbol{X}'\boldsymbol{X}) = \sigma^2\boldsymbol{R}^{-1}(\boldsymbol{R}^{-1})'$$

The R function `chol2inv` calculates $\boldsymbol{R}^{-1}(\boldsymbol{R}^{-1})'$ directly from $\boldsymbol{R}$ without evaluating $\boldsymbol{R}^{-1}$ explicitly (not a big deal in most cases but when p is very large it should be faster and more accurate than evaluating $\boldsymbol{R}^{-1}$ explicitly).

Also, the determinant of $\boldsymbol{X}'\boldsymbol{X}$ is

$$|\boldsymbol{X}'\boldsymbol{X}| = |\boldsymbol{R}'\boldsymbol{R}| = |\boldsymbol{R}|^2 = \left(\prod_{i=1}^{p} r_{i,i}\right)^2$$

The fitted values $\widehat{\boldsymbol{y}}$ are $\boldsymbol{Q}_1\boldsymbol{Q}_1'\boldsymbol{y}$ and thus the *hat matrix* (which puts a "hat" on $\boldsymbol{y}$ by transforming it to $\widehat{\boldsymbol{y}}$) is the $n \times n$ matrix $\boldsymbol{Q}_1\boldsymbol{Q}_1'$. Often we are interested in the diagonal elements of the hat matrix, which are the sums of the squares of rows of $\boldsymbol{Q}_1$. (In practice you don't want to calculate the entire $n \times n$ hat matrix just to get the diagonal elements when n could be very large.)

The residuals, $\widehat{\boldsymbol{e}} = \boldsymbol{y} - \widehat{\boldsymbol{y}}$, are calculated as $\widehat{\boldsymbol{e}} = \boldsymbol{Q}_2\boldsymbol{Q}_2'\boldsymbol{y}$.

The matrices $\boldsymbol{Q}_1\boldsymbol{Q}_1'$ and $\boldsymbol{Q}_2\boldsymbol{Q}_2'$ are *projection matrices*, which means that they are symmetric and *idempotent*. (A square matrix $\boldsymbol{A}$ is idempotent if $\boldsymbol{A}\boldsymbol{A} = \boldsymbol{A}$.) When $\operatorname{rank}(\boldsymbol{X}) = p$, the hat matrix $\boldsymbol{Q}_1\boldsymbol{Q}_1'$ projects any vector in $\mathbb{R}^n$ onto the column span of $\boldsymbol{X}$. The other projection, $\boldsymbol{Q}_2\boldsymbol{Q}_2'$, is onto the subspace orthogonal to the column span of $\boldsymbol{X}$ (see the figure on the front cover of the text).

R Exercise: We have already seen that $\widehat{\boldsymbol{\beta}}$ can be calculated as

```
> backsolve(R, crossprod(Q1, y))
```

```
            [,1]
[1,] 0.005085714
[2,] 0.876285714
```

or as

```
> qr.coef(qrlm1, y)
```

```
(Intercept)        carb
0.005085714 0.876285714
```

The functions `qr.fitted()` and `qr.fitted()` perform projection onto $\operatorname{col}(\boldsymbol{X})$ and onto its orthogonal complement, respectively.

The fitted values are, unsurprisingly, calculated as

```
> qr.fitted(qrlm1, y)
```

```
         1          2          3          4          5          6
0.09271429 0.26797143 0.44322857 0.53085714 0.61848571 0.79374286
```

```
> all.equal(qr.fitted(qrlm1, y), fitted(lm1))
```

```
[1] TRUE
```

and the residuals as

```
> qr.resid(qrlm1, y)
```

```
           1            2            3            4            5            6
-0.006714286  0.001028571  0.002771429  0.007142857  0.007514286 -0.011742857
```

```
> all.equal(qr.resid(qrlm1, y), residuals(lm1))
```

```
[1] TRUE
```

We use the explicit calculations for illustration only. In practice, use of the “extractor” methods, `fitted()` and `residuals()`, is preferred.

If we wanted the projection matrices $\boldsymbol{P}_1$ for projection onto $\operatorname{col}\boldsymbol{X}$ and onto the residual space we could form them as

```
> (P1 <- tcrossprod(Q1))

            [,1]        [,2]      [,3]       [,4]        [,5]        [,6]
[1,]  0.59183673  0.38775510 0.1836735 0.08163265 -0.02040816 -0.22448980
[2,]  0.38775510  0.28163265 0.1755102 0.12244898  0.06938776 -0.03673469
[3,]  0.18367347  0.17551020 0.1673469 0.16326531  0.15918367  0.15102041
[4,]  0.08163265  0.12244898 0.1632653 0.18367347  0.20408163  0.24489796
[5,] -0.02040816  0.06938776 0.1591837 0.20408163  0.24897959  0.33877551
[6,] -0.22448980 -0.03673469 0.1510204 0.24489796  0.33877551  0.52653061
```

and

```
> (P2 <- tcrossprod(Q[, -(1:2)]))

            [,1]        [,2]       [,3]        [,4]        [,5]        [,6]
[1,]  0.40816327 -0.38775510 -0.1836735 -0.08163265  0.02040816  0.22448980
[2,] -0.38775510  0.71836735 -0.1755102 -0.12244898 -0.06938776  0.03673469
[3,] -0.18367347 -0.17551020  0.8326531 -0.16326531 -0.15918367 -0.15102041
[4,] -0.08163265 -0.12244898 -0.1632653  0.81632653 -0.20408163 -0.24489796
[5,]  0.02040816 -0.06938776 -0.1591837 -0.20408163  0.75102041 -0.33877551
[6,]  0.22448980  0.03673469 -0.1510204 -0.24489796 -0.33877551  0.47346939
```

respectively (the expression `Q[, -(1:2)]` drops the first two columns of $\boldsymbol{Q}$ producing $\boldsymbol{Q}_2$). (And, of course, we don't do this in practice, especially if n is large. Instead we use `qr.fitted` or `qr.resid` if we want to project vectors other than $\boldsymbol{y}$.)

We can check that `P1` and `P2` are projection matrices. They are obviously symmetric by construction so we check the idempotent property

```
> all.equal(P1 %*% P1, P1)

[1] TRUE

> all.equal(P2 %*% P2, P2)

[1] TRUE
```

Because `P1` is projection onto col($\boldsymbol{X}$) it should take $\boldsymbol{X}$ to itself

```
> all.equal(P1 %*% X, X, check.attr=FALSE)

[1] TRUE
```

and `P2` should take $\boldsymbol{X}$ to zeros, although in practice we expect very small but possibly non-zero values.

```
> all.equal(P2 %*% X, 0 * X, check.attr=FALSE)

[1] TRUE
```

(The weird construction, `0 * X`, create a matrix of zeros that is the same size as $\boldsymbol{X}$.)

Because `P1` is the hat matrix, we can get its diagonal elements as

```
> diag(P1)
```

```
[1] 0.5918367 0.2816327 0.1673469 0.1836735 0.2489796 0.5265306
```

As mentioned, the alternative calculation is

```
> rowSums(Q1^2)
```

```
[1] 0.5918367 0.2816327 0.1673469 0.1836735 0.2489796 0.5265306
```

and a third way, preferred in practice, is

```
> hatvalues(lm1)
```

```
        1         2         3         4         5         6
0.5918367 0.2816327 0.1673469 0.1836735 0.2489796 0.5265306
```

rank($\boldsymbol{X}$), which is the number of linearly independent columns in $\boldsymbol{X}$, is calculated during the decomposition and also stored as the `$rank` component of the fitted model

```
> lm1$rank
```

```
[1] 2
```

```
> qrlm1$rank
```

```
[1] 2
```

1.3.4 R functions related to the QR decomposition

To review, every time you fit a linear model with `lm` or `aov` or `lm.fit`, the returned object contains a `$qr` component. This is a condensed form of the QR decomposition of $\boldsymbol{X}$, only slightly larger than $\boldsymbol{X}$ itself. Its class is `"qr"`.

There are several extractor functions for a `"qr"` object: `qr.R()`. `qr.Q()` and `qr.X()`, which regenerates the original matrix. By default `qr.Q()` returns the matrix called $\boldsymbol{Q}_1$ above with p columns but you can specify the number of columns desired. Typical alternative choices are n or rank($\boldsymbol{X}$).

The `$rank` component of a `"qr"` object is the computed rank of $\boldsymbol{X}$ (and, hence, of $\boldsymbol{R}$). The `$pivot` component is the permutation applied to the columns. It will be `1:p` when rank($\boldsymbol{X}$) $= p$ but when rank($\boldsymbol{X}$) $< p$ it may be other than the identity permutation.

Several functions are applied to a `"qr"` object and a vector or matrix. These include `qr.coef()`, `qr.qy()`, `qr.qty()`, `qr.resid()` and `qr.fitted()`. The `qr.qy()` and `qr.qty()` functions multiply an n-vector or an $n \times m$ matrix by $\boldsymbol{Q}$ or $\boldsymbol{Q}'$ without ever forming $\boldsymbol{Q}$. Similarly, `qr.fitted()` creates $\boldsymbol{Q}_1\boldsymbol{Q}_1'\boldsymbol{x}$ and `qr.resid()` creates $\boldsymbol{Q}_2\boldsymbol{Q}_2'\boldsymbol{x}$ without forming $\boldsymbol{Q}$.

The `is.qr()` function tests an object to determine if it is of class `"qr"`.

1.4 Related matrix decompositions

1.4.1 The Cholesky decomposition

The Cholesky decomposition of a positive definite symmetric matrix, which means a $p \times p$ symmetric matrix $\boldsymbol{A}$ such that $\boldsymbol{x}'\boldsymbol{A}\boldsymbol{x} > 0$ for all non-zero $\boldsymbol{x} \in \mathbb{R}^p$ is of the form

$$\boldsymbol{A} = \boldsymbol{R}'\boldsymbol{R} = \boldsymbol{L}\boldsymbol{L}'$$

where $\boldsymbol{R}$ is an upper triangular $p \times p$ matrix and $\boldsymbol{L} = \boldsymbol{R}'$ is lower triangular. The two forms are the same decomposition: it is just a matter of whether you want $\boldsymbol{R}$, the factor on the right, or $\boldsymbol{L}$, the factor on the left. Generally statisticians write the decomposition as $\boldsymbol{R}'\boldsymbol{R}$.

The decomposition is only determined up to changes in sign of the rows of $\boldsymbol{R}$ (or, equivalently, the columns of $\boldsymbol{L}$). For definiteness we require positive diagonal elements in $\boldsymbol{R}$.

When $\operatorname{rank}(\boldsymbol{X}) = p$ the Cholesky decomposition $\boldsymbol{R}$ of $\boldsymbol{X}'\boldsymbol{X}$ is the equal to the matrix $\boldsymbol{R}$ from the QR decomposition up to changes in sign of rows. The matrix $\boldsymbol{X}'\boldsymbol{X}$ matrix is obviously symmetric and it is positive definite because

$$\boldsymbol{x}'(\boldsymbol{X}'\boldsymbol{X})\boldsymbol{x} = \boldsymbol{x}'(\boldsymbol{R}'\boldsymbol{R})\boldsymbol{x} = \|\boldsymbol{R}\boldsymbol{x}\|^2 \geq 0$$

with equality only when $\boldsymbol{R}\boldsymbol{x} = \boldsymbol{0}$, which, when $\operatorname{rank}(\boldsymbol{R}) = p$, implies that $\boldsymbol{x} = \boldsymbol{0}$.

1.4.2 Evaluation of the Cholesky decomposition

The R function `chol()` evaluates the Cholesky decomposition. As mentioned above `chol2inv()` creates $(\boldsymbol{X}'\boldsymbol{X})^{-1}$ directly from the Cholesky decomposition of $\boldsymbol{X}'\boldsymbol{X}$.

Generally the QR decomposition is preferred to the Cholesky decomposition for least squares problems because there is a certain loss of precision when forming $\boldsymbol{X}'\boldsymbol{X}$. However, when n is very large you may want to build up $\boldsymbol{X}'\boldsymbol{X}$ using blocks of rows. Also, if $\boldsymbol{X}$ is *sparse* it is an advantage to use sparse matrix techniques to evaluate and store the Cholesky decomposition.

The `Matrix` package for R provides even more capabilities related to the Cholesky decomposition, especially for sparse matrices.

For everything we will do in Statistics 849 the QR decomposition should be the method of choice.

R Exercises:

```
> chol(crossprod(X))
```

```
            (Intercept)      carb
(Intercept)    2.449490 1.2655697
carb           0.000000 0.6390097
```

1.4.3 The singular value decomposition

Another decomposition related to orthogonal matrices is the singular value decomposition (or SVD) in which the matrix $\boldsymbol{X}$ is reduced to a diagonal form

$$\boldsymbol{X} = \boldsymbol{U}_1\boldsymbol{D}\boldsymbol{V}' = \boldsymbol{U}\begin{bmatrix}\boldsymbol{D}\\ \boldsymbol{0}\end{bmatrix}\boldsymbol{V}' \tag{1.11}$$

where $\boldsymbol{U}$ is an $n \times n$ orthogonal matrix, $\boldsymbol{D}$ is a $p \times p$ diagonal matrix with non-negative diagonal elements (which are called the *singular values* of $\boldsymbol{X}$) and $\boldsymbol{V}$ is a $p \times p$ orthogonal matrix. As for $\boldsymbol{Q}$ and $\boldsymbol{Q}_1$, $\boldsymbol{U}_1$ consists of the first p columns of $\boldsymbol{U}$. For definiteness we order the diagonal elements of $\boldsymbol{D}$, which must be non-negative, in decreasing order.

Just like $\boldsymbol{Q}_1$, the columns of $\boldsymbol{U}_1$ form an orthonormal basis for $\operatorname{col}(\boldsymbol{X})$ when $\boldsymbol{X}$ has full column rank (which means that the singular values are all safely positive). If $\operatorname{rank}(X) = r < p$ then the first r columns of $\boldsymbol{U}$ form the orthonormal basis.

One way to visualize the singular value decomposition of $\boldsymbol{X}$ is to remember that a p-sphere in $\mathbb{R}^p$ will get mapped to an ellipsoid in $\operatorname{col}(\boldsymbol{X})$ by $\boldsymbol{X}$. The singular values are the lengths of the principal axes of this ellipsoid. The right singular vectors (columns of $\boldsymbol{V}$) are the directions in the parameter space that map onto the principal axes of the ellipsoid. The first $\operatorname{rank}(\boldsymbol{X})$ left singular vectors (columns of $\boldsymbol{U}$) are the principal axes of the ellipsoid.

The singular value decomposition of $\boldsymbol{X}$ is related to the eigendecomposition or spectral decomposition of $\boldsymbol{X}'\boldsymbol{X}$ because

$$\boldsymbol{X}'\boldsymbol{X} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{U}_1'\boldsymbol{U}_1\boldsymbol{D}\boldsymbol{V}' = \boldsymbol{V}\boldsymbol{D}^2\boldsymbol{V}'$$

implying that the eigenvalues of $\boldsymbol{X}'\boldsymbol{X}$ are the squares of the singular values of $\boldsymbol{X}$ and the right singular vectors, which are the columns of $\boldsymbol{V}$, are also the eigenvectors of $\boldsymbol{X}'\boldsymbol{X}$

Calculation of the SVD is an iterative (as opposed to a direct) computation and potentially more computing intensive than the QR decomposition, although modern methods for evaluating the SVD are very good indeed.

Symbolically we can write the least squares solution in the full-rank case as

$$\widehat{\boldsymbol{\beta}} = \boldsymbol{V}\boldsymbol{D}^{-1}\boldsymbol{U}_1'\boldsymbol{y}$$

where $\boldsymbol{D}^{-1}$ is a diagonal matrix whose diagonal elements are the inverses of the diagonal elements of $\boldsymbol{D}$.

The pseudo-inverse or *generalized inverse* of $\boldsymbol{X}$, written $\boldsymbol{X}^-$, is calculated from the pseudo-inverse of the diagonal matrix, $\boldsymbol{D}$. In theory the diagonal elements of $\boldsymbol{D}^-$ are $1/d_{i,i}$ when $d_{i,i} \neq 0$ and 0 when $d_{i,i} = 0$. However, we can't count on $d_{i,i}$ being 0 even when, in theory, it should be. We need to decide when the singular values are close to zero, which is actually a very difficult problem. At best we can use some heuristics, based on the ratio of $d_{i,i}/d_{1,1}$, to decide when a diagonal element is "effectively zero".

The use of the pseudo-inverse seems to be a convenient way to handle rank-deficient $\boldsymbol{X}$ matrices but, as mentioned above, the best way to handle rank-deficient $\boldsymbol{X}$ matrices is not to produce them in the first place. Even when a rank-deficient $\boldsymbol{X}$ is produced we use a pivoted QR decomposition rather than a pseudo-inverse.

R Exercises: The SVD of our model matrix $\boldsymbol{X}$ is

```
> str(Xsv <- svd(X))

List of 3
 $ d: num [1:2] 2.773 0.564
 $ u: num [1:6, 1:2] 0.334 0.368 0.403 0.42 0.437 ...
 $ v: num [1:2, 1:2] 0.878 0.479 -0.479 0.878
```

We see that, by default, the `svd()` function produces the diagonal of $\boldsymbol{D}$, the matrix $\boldsymbol{U}_1$ and the matrix $\boldsymbol{V}$. We should check that $\boldsymbol{X} = \boldsymbol{U}_1\boldsymbol{D}\boldsymbol{V}'$, as advertised. We could form a diagonal matrix $\boldsymbol{D}$ from the `$d` component of `Xsv` but multiplication of a matrix on the left by a diagonal matrix corresponds to scaling its rows, so we write the reconstruction as

```
> Xsv$u %*% (Xsv$d * t(Xsv$v))

     [,1] [,2]
[1,]    1  0.1
[2,]    1  0.3
[3,]    1  0.5
[4,]    1  0.6
[5,]    1  0.7
[6,]    1  0.9

> all.equal(Xsv$u %*% (Xsv$d * t(Xsv$v)), X, check.attr=FALSE)

[1] TRUE
```

We check that the matrix $\boldsymbol{U}_1$ has orthonormal columns and that $\boldsymbol{V}$ is orthogonal

```
> zapsmall(crossprod(Xsv$u))

     [,1] [,2]
[1,]    1    0
[2,]    0    1

> zapsmall(crossprod(Xsv$v))

     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

The squares of the singular values should be the eigenvalues of $\boldsymbol{X}'\boldsymbol{X}$ and the eigenvectors of $\boldsymbol{X}'\boldsymbol{X}$ should be the columns of $\boldsymbol{V}$, up to changes in sign along columns. (The eigenvectors, which are really just directions, are only determined up to changes in sign, and in the case of repeated eigenvalues, only up to orthogonal transformation within the repeated eigenvalue's eigenspace.)

```
> str(ev <- eigen(crossprod(X)))
```

```
List of 2
 $ values : num [1:2] 7.691 0.319
 $ vectors: num [1:2, 1:2] -0.878 -0.479 0.479 -0.878

> Xsv$d^2

[1] 7.6914651 0.3185349

> all.equal(ev$values, Xsv$d^2)

[1] TRUE

> ev$vectors

           [,1]       [,2]
[1,] -0.8778294  0.4789735
[2,] -0.4789735 -0.8778294

> Xsv$v

          [,1]       [,2]
[1,] 0.8778294 -0.4789735
[2,] 0.4789735  0.8778294

> all.equal(-Xsv$v, ev$vectors)

[1] TRUE
```

In practice, you never need to calculate the eigenvalues and eigenvectors of $\boldsymbol{X}'\boldsymbol{X}$. It is more effective and more stable to calculate the singular value decomposition of $\boldsymbol{X}$ and use the squares of the singular values and the `$v` component (assuming that you really do need the eigenvalues and eigenvectors which, most of the time, you don't).

The reason that it is preferable to work with decompositions of $\boldsymbol{X}$ rather than forming $\boldsymbol{X}'\boldsymbol{X}$ is related to the condition number of these matrices. As described on the Wikipedia page, the condition number of a matrix, written $\kappa(\boldsymbol{X})$, is the ratio of its largest and smallest singular values. Obviously we must have $\kappa(\boldsymbol{X}) \geq 1$. A matrix with κ close to 1 is well-conditioned. A matrix with a very large condition number is close to being singular, in that spheres are mapped to highly elongated ellipsoids.

An orthogonal matrix or a rectangular matrix with orthonormal columns must have a condition number of 1 because it maps a sphere to a sphere. (Recall that, for us, rectangular matrices like $\boldsymbol{X}$ have more rows than columns. In the opposite case, more columns than rows, it would be the rows that are orthonormal.) In fact, all the singular values of an orthogonal matrix must be unity because it preserves lengths so the unit sphere gets mapped to the unit sphere.

We can check that matrices like $\boldsymbol{Q}$, $\boldsymbol{Q}_1$ and $\boldsymbol{U}_1$ have a condition number of 1.

```
> svd(Q, nu=0, nv=0)$d
```

```
[1] 1 1 1 1 1 1

> kappa(Q)

[1] 1

> svd(Q1, nu=0, nv=0)$d

[1] 1 1

> kappa(Q1)

[1] 1

> kappa(Xsv$u)

[1] 1
```

The condition number of $\boldsymbol{X}$ can be explicitly calculated as

```
> Xsv$d

[1] 2.773349 0.564389

> (kappaX <- Xsv$d[1]/Xsv$d[length(Xsv$d)])

[1] 4.913897
```

(The complicated expression in the last line is to generalize the calculation. It will give the correct answer when there are more than two singular values.) The `kappa()` function, by default, produces an upper bound on the condition number, because this upper bound can be calculated directly. To get the exact value set the optional argument `exact=TRUE`.

```
> kappa(X)

[1] 5.1073

> kappa(X, exact=TRUE)

[1] 4.913897
```

In practice we usually calculate the reciprocal of the condition number because its value is in $[0, 1]$ and it is easier to decide when it is close to zero instead of trying to decide when $\kappa(\boldsymbol{X})$ is "close to" ∞. We compare the reciprocal condition number to the relative machine precision,

```
> .Machine$double.eps

[1] 2.220446e-16
```

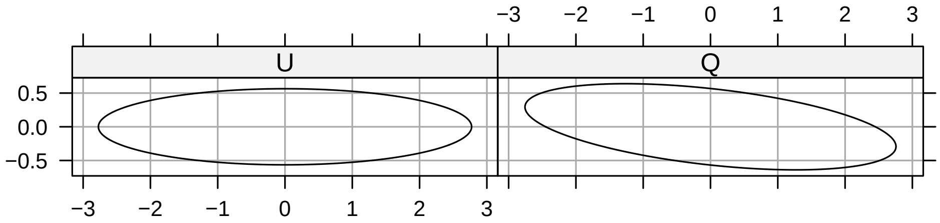

Figure 1.1: The image of the unit circle in $\mathbb{R}^2$ after mapping by $\boldsymbol{U}_1'\boldsymbol{X}$ (left panel) and by $\boldsymbol{Q}_1'\boldsymbol{X}$ (right panel)

A matrix is considered computationally singular when its reciprocal condition number is within some multiple, typical values are 10 or 100, of this number.

Getting back to the question of why we prefer to work with $\boldsymbol{X}$ directly, instead of forming $\boldsymbol{X}'\boldsymbol{X}$, it is because $\kappa(\boldsymbol{X}'\boldsymbol{X}) = \kappa(\boldsymbol{X})^2$. If $\kappa(\boldsymbol{X}) = 10^6$, which is large but not catastrophically so, then $\kappa(\boldsymbol{X}'\boldsymbol{X})$ will be 10^{12}, which means it is very close to being singular.

Finally, let's revisit the idea of the singular values being the lengths of the principal axes of the image of the unit sphere in the map $\boldsymbol{\beta} \to \boldsymbol{X}\boldsymbol{\beta}$. When $p = 2$ the unit sphere is the circle of radius 1 centered at the origin and the ellipsoid mentioned above will be an ellipse.

A convenient way of creating a $2 \times N$ matrix whose columns are the points on the unit circle is to start with a sequence of values from 0 to 2π and use its sines and cosines

```
> str(rad <- seq(0, 2*pi, len=201))

 num [1:201] 0 0.0314 0.0628 0.0942 0.1257 ...

> str(circ <- rbind(cos(rad), sin(rad)))

 num [1:2, 1:201] 1 0 0.9995 0.0314 0.998 ...
```

The n-dimensional response vectors corresponding to these points on the circle are

```
> fits <- X %*% circ
```

To plot the this image in two dimensions (Fig. 1.1) we need to represent these points with respect to an orthogonal basis for col($\boldsymbol{X}$). Fortunately we have two such bases: the columns of $\boldsymbol{Q}_1$ and of $\boldsymbol{U}_1$. In the $\boldsymbol{U}_1$ basis the principal axes of the ellipse correspond to the coordinate axes. In the $\boldsymbol{Q}_1$ basis the principal axes are skewed.

1.5 Theoretical results on the eigendecomposition

1.5.1 Eigenvalues and Eigenvectors

For any $k \times k$ matrix $\boldsymbol{A}$, the roots of the k^{th} degree polynomial equation in λ, $|\lambda \boldsymbol{I}_k - \boldsymbol{A}| = 0$, which we will write as $\lambda_1, \ldots, \lambda_k$ are called the *eigenvalues* of $\boldsymbol{A}$. The polynomial is called the *characteristic polynomial* of $\boldsymbol{A}$.

Any nonzero $n \times 1$ vector $\boldsymbol{v}_i \neq \boldsymbol{0}$ such that $\boldsymbol{A}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$ is an *eigenvector* of $\boldsymbol{A}$ corresponding to the eigenvalue λ_i.

For any diagonal matrix $\boldsymbol{D} = \text{diag}(d_1, \dots, d_k)$, $|\lambda \boldsymbol{I}_k - \boldsymbol{D}| = \prod_{i=1}^{k}(\lambda - d_i) = 0$ has roots d_i, therefore the diagonal elements $d_i,\ i = 1, \dots, n$ are the eigenvalues of $\boldsymbol{D}$

If $\boldsymbol{Q}$ is an orthogonal matrix, then $\boldsymbol{QAQ}'$ and $\boldsymbol{A}$ have the same eigenvalues.

Proof.

$$\begin{aligned} |\lambda \boldsymbol{I} - \boldsymbol{QAQ}'| &= |\lambda \boldsymbol{QQ}' - \boldsymbol{QAQ}'| \\ &= |\boldsymbol{Q}||\lambda \boldsymbol{Q}' - \boldsymbol{AQ}'| \\ &= |\boldsymbol{Q}||\lambda \boldsymbol{I} - \boldsymbol{A}||\boldsymbol{Q}'| \\ &= |\boldsymbol{Q}|^2 |\lambda \boldsymbol{I} - \boldsymbol{A}| = 1|\lambda \boldsymbol{I} - \boldsymbol{A}| \\ &= |\lambda \boldsymbol{I} - \boldsymbol{A}| \end{aligned} \tag{1.12}$$

□

Note: Although the eigenvalues are defined as the roots of the characteristic polynomial, in practice they are not calculated this way. In fact, if you check the documentation for function `solve.polynomial()` in the `polynom` package for R you will find that it uses the numerical methods for evaluating the eigenvalues of the *companion matrix* of a polynomial to solve for the polynomial's roots.

1.5.2 Diagonalization of a Symmetric Matrix

For any $k \times k$ symmetric matrix $\boldsymbol{A}$ (i.e. $\boldsymbol{A}' = \boldsymbol{A}$), there exists an orthogonal matrix $\boldsymbol{Q}$ such that $\boldsymbol{QAQ}'$ is a diagonal matrix $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_n)$ where λ_i are the eigenvalues of $\boldsymbol{A}$. The corresponding eigenvectors of $\boldsymbol{A}$ are the column vectors of $\boldsymbol{Q}$

Proof. Let $\boldsymbol{e}_i,\ i = 1, \dots, n$ be $n \times 1$ unit vectors that form the canonical basis of $\mathbb{R}^n$, (i.e. $\boldsymbol{e}_i = (0, \dots, 1, \dots, 0)'$ where the 1 is in the i^{th} position) and $\boldsymbol{q}_i$ be the ith column of $\boldsymbol{Q}$. That is, $\boldsymbol{q}_i = \boldsymbol{Q}\boldsymbol{e}_i$. Then

$$\boldsymbol{Q}'\boldsymbol{AQ} = \boldsymbol{\Lambda} \Rightarrow \boldsymbol{Q}'\boldsymbol{AQ}\boldsymbol{e}_i = \boldsymbol{\Lambda} e_i = \lambda_i \boldsymbol{e}_i$$

Multiplying on the left by $\boldsymbol{Q}$ produces

$$\boldsymbol{A}\boldsymbol{q}_i = \underbrace{\boldsymbol{QQ}'}_{\boldsymbol{I}}\boldsymbol{AQ}\boldsymbol{e}_i = \lambda_i \boldsymbol{Q}\boldsymbol{e}_i = \lambda_1 \boldsymbol{q}_i.$$

□

1.5.3 Spectral Decomposition

From the relationship $\boldsymbol{Q}'\boldsymbol{A}\boldsymbol{Q} = \boldsymbol{\Lambda}$ just established for a $k \times k$ symmetric matrix $\boldsymbol{A}$ we can compute its spectral decomposition,

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}' = \sum_{i=1}^{k} \lambda_i \boldsymbol{q}_i \boldsymbol{q}_i'$$

where $\boldsymbol{q}_i$ is the i^{th} column of $\boldsymbol{Q}$.

$$\boldsymbol{Q}\boldsymbol{Q}' = \sum_{i=1}^{k} \boldsymbol{q}_i \boldsymbol{q}_i' = \boldsymbol{I}$$

1.5.4 Trace and Determinant of A

The relationship $\boldsymbol{Q}'\boldsymbol{A}\boldsymbol{Q} = \boldsymbol{\Lambda}$ for symmetric $\boldsymbol{A}$ implies that the trace, $\operatorname{tr}(\boldsymbol{A})$, and the determinant, $|\boldsymbol{A}|$, are the same as those of $\boldsymbol{\Lambda}$.

$$\operatorname{tr}(\boldsymbol{A}) = \operatorname{tr}(\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}') = \operatorname{tr}(\boldsymbol{\Lambda}\boldsymbol{Q}\boldsymbol{Q}') = \operatorname{tr}(\boldsymbol{\Lambda}) = \sum_{i=1}^{k} \lambda_i$$

where we have used the property that $\operatorname{tr}(\boldsymbol{C}\boldsymbol{D}) = \operatorname{tr}(\boldsymbol{D})$ for any conformable matrices $\boldsymbol{C}$ and $\boldsymbol{D}$ (meaning that if $\boldsymbol{C}$ is $m \times n$ then $\boldsymbol{D}$ must be $n \times m$).

$$|\boldsymbol{A}| = |\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}'| = |\boldsymbol{Q}||\boldsymbol{\Lambda}||\boldsymbol{Q}'| = |\boldsymbol{Q}|^2|\boldsymbol{\Lambda}| = \prod_{i=1}^{k} \lambda_i$$

Chapter 2

Quadratic Forms of Random Variables

2.1 Quadratic Forms

For a $k \times k$ symmetric matrix $\boldsymbol{A} = \{a_{ij}\}$ the quadratic function of k variables $\boldsymbol{x} = (x_1, \ldots, x_n)'$ defined by

$$Q(\boldsymbol{x}) = \boldsymbol{x}'\boldsymbol{A}\boldsymbol{x} = \sum_{i=1}^{k}\sum_{j=1}^{k} a_{i,j}x_i x_j$$

is called the *quadratic form* with matrix $\boldsymbol{A}$.

If $\boldsymbol{A}$ is not symmetric, we can have an equivalent expression/quadratic form replacing $\boldsymbol{A}$ by $(\boldsymbol{A} + \boldsymbol{A}')/2$.

Definition 1. *$Q(\boldsymbol{x})$ and the matrix $\boldsymbol{A}$ are called* positive definite *if*

$$Q(\boldsymbol{x}) = \boldsymbol{x}'\boldsymbol{A}\boldsymbol{x} > 0, \quad \forall\, \boldsymbol{x} \in \mathbb{R}^k,\ \boldsymbol{x} \neq \boldsymbol{0}$$

and positive semi-definite *if*

$$Q(\boldsymbol{x}) \geq \forall\, \boldsymbol{x} \in \mathbb{R}^k$$

For negative definite *and* negative semi-definite*, replace the $>$ and $\geq$ in the above definitions by $<$ and $\leq$, respectively.*

Theorem 1. *A symmetric matrix $\boldsymbol{A}$ is positive definite if and only if it has a Cholesky decomposition $\boldsymbol{A} = \boldsymbol{R}'\boldsymbol{R}$ with strictly positive diagonal elements in $\boldsymbol{R}$, so that $\boldsymbol{R}^{-1}$ exists. (In practice this means that none of the diagonal elements of $\boldsymbol{R}$ are very close to zero.)*

Proof. The "if" part is proven by construction. The Cholesky decomposition, $\boldsymbol{R}$, is constructed a row at a time and the diagonal elements are evaluated as the square roots of expressions calculated from the current row of $\boldsymbol{A}$ and previous rows of $\boldsymbol{R}$. If the expression whose square root is to be calculated is not positive then you can determine a non-zero $\boldsymbol{x} \in \mathbb{R}^k$ for which $\boldsymbol{x}'\boldsymbol{A}\boldsymbol{x} \leq 0$.

Suppose that $\boldsymbol{A} = \boldsymbol{R}'\boldsymbol{R}$ with $\boldsymbol{R}$ invertible. Then

$$\boldsymbol{x}'\boldsymbol{A}\boldsymbol{x} = \boldsymbol{x}'\boldsymbol{R}'\boldsymbol{R}\boldsymbol{x} = \|\boldsymbol{R}\boldsymbol{x}\|^2 \geq 0$$

with equality only if $\boldsymbol{R}\boldsymbol{x} = \boldsymbol{0}$. But if $\boldsymbol{R}^{-1}$ exists then $\boldsymbol{x} = \boldsymbol{R}^{-1}\boldsymbol{0}$ must also be zero. □

Transformation of Quadratic Forms:

Theorem 2. *Suppose that $\boldsymbol{B}$ is a $k \times k$ nonsingular matrix. Then the quadratic form $Q^*(\boldsymbol{y}) = \boldsymbol{y}'\boldsymbol{B}'\boldsymbol{A}\boldsymbol{B}\boldsymbol{y}$ is positive definite if and only if $Q(\boldsymbol{x}) = \boldsymbol{x}'\boldsymbol{A}\boldsymbol{x}$ is positive definite. Similar results hold for positive semi-definite, negative definite and negative semi-definite.*

Proof.

$$Q^*(\boldsymbol{y}) = \boldsymbol{y}'\boldsymbol{B}'\boldsymbol{A}\boldsymbol{B}\boldsymbol{y} = \boldsymbol{x}'\boldsymbol{A}\boldsymbol{x} > 0$$

where $\boldsymbol{x} = \boldsymbol{B}\boldsymbol{y} \neq \boldsymbol{0}$ because $\boldsymbol{y} \neq \boldsymbol{0}$ and $\boldsymbol{B}$ is nonsingular. □

Theorem 3. *For any $k \times k$ symmetric matrix $\boldsymbol{A}$ the quadratic form defined by $\boldsymbol{A}$ can be written using its spectral decomposition as*

$$Q(\boldsymbol{x}) = \boldsymbol{x}'\boldsymbol{A}\boldsymbol{x} = \sum_{i=1}^{k} \lambda_i \|\boldsymbol{q}_i'\boldsymbol{x}\|^2$$

where the eigendecomposition of of $\boldsymbol{A}$ is $\boldsymbol{Q}'\boldsymbol{\Lambda}\boldsymbol{Q}$ with $\boldsymbol{\Lambda}$ diagonal with diagonal elements λ_i, $i = 1, \ldots, k$, $\boldsymbol{Q}$ is the orthogonal matrix with the eigenvectors, $\boldsymbol{q}_i$, $i = 1, \ldots, k$ as its columns. (Be careful to distinguish the bold face $\boldsymbol{Q}$, which is a matrix, from the unbolded $Q(\boldsymbol{x})$, which is the quadratic form.)

Proof. For any $\boldsymbol{x} \in \mathbb{R}^k$ let $\boldsymbol{y} = \boldsymbol{Q}'\boldsymbol{x} = \boldsymbol{Q}^{-1}\boldsymbol{x}$. Then

$$Q(\boldsymbol{x}) = \mathrm{tr}(\boldsymbol{x}'\boldsymbol{A}\boldsymbol{x}) = \mathrm{tr}(\boldsymbol{x}'\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}'\boldsymbol{x}) = \mathrm{tr}(\boldsymbol{y}'\boldsymbol{\Lambda}\boldsymbol{y}) = \mathrm{tr}(\boldsymbol{\Lambda}\boldsymbol{y}\boldsymbol{y}') == \sum_{i=1}^{k} \lambda_i y_i^2 = \sum_{i=1}^{k} \lambda_i \|\boldsymbol{q}_i'\boldsymbol{x}\|^2$$

This proof uses a common "trick" of expressing the scalar $Q(\boldsymbol{x})$ as the trace of a 1×1 matrix so we can reverse the order of some matrix multiplications. □

Corollary 1. *A symmetric matrix $\boldsymbol{A}$ is positive definite if and only if its eigenvalues are all positive, negative definite if and only if its eignevalues are all negative, and positive semi-definite if all its eigenvalues are non-negative.*

Corollary 2. $\mathrm{rank}(\boldsymbol{A}) = \mathrm{rank}(\boldsymbol{\Lambda})$ *hence* $\mathrm{rank}(\boldsymbol{A})$ *equals the number of non-zero eigenvalues of* $\boldsymbol{A}$

2.2 Idempotent Matrices

Definition 2 (Idempotent)**.** *The $k \times k$ matrix $\boldsymbol{A}$, is* idempotent *if $\boldsymbol{A}^2 = \boldsymbol{A}\,\boldsymbol{A} = \boldsymbol{A}$.*

Definition 3 (Projection matrices)**.** *A symmetric, idempotent matrix $\boldsymbol{A}$ is a* projection matrix. *The effect of the mapping $\boldsymbol{x} \to \boldsymbol{A}\boldsymbol{x}$ is orthogonal projection of $\boldsymbol{x}$ onto* $\mathrm{col}(A)$.

Theorem 4. *All the eigenvalues of an idempotent matrix are either zero or one.*

Proof. Suppose that λ is an eigenvalue of the idempotent matrix $\boldsymbol{A}$. Then there exists a non-zero $\boldsymbol{x}$ such that $\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x}$. But $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}\boldsymbol{A}\boldsymbol{x}$ because $\boldsymbol{A}$ is idempotent. Thus

$$\lambda\boldsymbol{x} = \boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}(\lambda\boldsymbol{x}) = \lambda(\boldsymbol{A}\boldsymbol{x}) = \lambda^2\boldsymbol{x}$$

and

$$\mathbf{0} = \lambda^2\boldsymbol{x} - \lambda\boldsymbol{x} = \lambda(\lambda - 1)\boldsymbol{x}$$

for some non-zero $\boldsymbol{x}$, which implies that $\lambda = 0$ or $\lambda = 1$. □

Corollary 3. *The $k \times k$ symmetric matrix $\boldsymbol{A}$ is idempotent of rank$(\boldsymbol{A}) = r$ iff $\boldsymbol{A}$ has r eigenvalues equal to 1 and $k - r$ eigenvalues equal to 0*

Proof. A matrix $\boldsymbol{A}$ with r eigenvalues of 1 and $k - r$ eigenvalues of zero has r non-zero eigenvalues and hence $\operatorname{rank}(\boldsymbol{A}) = r$. Because $\boldsymbol{A}$ is symmetric its eigendecomposition is $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}'$ for an orthogonal $\boldsymbol{Q}$ and a diagonal $\boldsymbol{\Lambda}$. Because the eigenvalues of $\boldsymbol{\Lambda}$ are the same as those of $\boldsymbol{A}$, they must be all zeros or ones. That is all the diagonal elements of $\boldsymbol{\Lambda}$ are zero or one. Hence $\boldsymbol{\Lambda}$ is idempotent, $\boldsymbol{\Lambda}\boldsymbol{\Lambda} = \boldsymbol{\Lambda}$, and

$$\boldsymbol{A}\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}'\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}' = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}' = \boldsymbol{A}$$

is also idempotent. □

Corollary 4. *For a symmetric idempotent matrix $\boldsymbol{A}$, we have tr$(\boldsymbol{A})$ = rank$(\boldsymbol{A})$, which is the dimension of* $\operatorname{col}(\boldsymbol{A})$*, the space into which $\boldsymbol{A}$ projects.*

2.3 Expected Values and Covariance Matrices of Random Vectors

An k-dimensional *vector-valued random variable* (or, more simply, a *random vector*), $\mathcal{X}$, is a k-vector composed of k scalar random variables

$$\mathcal{X} = (\mathcal{X}_1, \ldots, \mathcal{X}_k)'$$

If the expected values of the component random variables are $\mu_i = E(\mathcal{X}_i)$, $i = 1, \ldots, k$ then

$$E(\mathcal{X}) = \boldsymbol{\mu}_{\mathcal{X}} = (\mu_1, \ldots, \mu_k)'$$

Suppose that $\mathcal{Y} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_m)'$ is an m-dimensional random vector, then the *covariance* of $\mathcal{X}$ and $\mathcal{Y}$, written $\operatorname{Cov}(\mathcal{X}, \mathcal{Y})$ is

$$\boldsymbol{\Sigma}_{XY} = \operatorname{Cov}(\mathcal{X}, \mathcal{Y}) = E[(\mathcal{X} - \boldsymbol{\mu}_{\mathcal{X}})(\mathcal{Y} - \boldsymbol{\mu}_{\mathcal{Y}})']$$

The *variance-covariance* matrix of $\mathcal{X}$ is

$$\operatorname{Var}(\mathcal{X}) = \boldsymbol{\Sigma}_{XX} = E[(\mathcal{X} - \boldsymbol{\mu}_{\mathcal{X}})(\mathcal{X} - \boldsymbol{\mu}_{\S})$$

Suppose that $\boldsymbol{c}$ is a constant m-vector, $\boldsymbol{A}$ is a constant $m \times k$ matrix and $\mathcal{Z} = \boldsymbol{Z}\mathcal{X} + \boldsymbol{c}$ is a linear transformation of $\mathcal{X}$. Then

$$E(\mathcal{Z}) = \boldsymbol{A}E(\mathcal{X}) + \boldsymbol{c}$$

and

$$\mathrm{Var}(\mathcal{Z}) = \boldsymbol{A}\,\mathrm{Var}(\mathcal{X})\boldsymbol{A}'$$

If we let $\mathcal{W} = \boldsymbol{B}\mathcal{Y} + \boldsymbol{d}$ be a linear transformation of $\mathcal{Y}$ for suitably sized $\boldsymbol{B}$ and $\boldsymbol{d}$ then

$$\mathrm{Cov}(\mathcal{Z}, \mathcal{W}) = \boldsymbol{A}\,\mathrm{Cov}(\mathcal{X}, \mathcal{Y})B'$$

Theorem 5. *The variance-covariance matrix $\boldsymbol{\Sigma}_{\mathcal{X},\mathcal{X}}$ of $\mathcal{X}$ is a symmetric and positive semi-definite matrix*

Proof. The result follows from the property that the variance of a scalar random variable is nonnegative. Suppose that $\boldsymbol{b}$ is any nonzero, constant k-vector. Then

$$0 \leq \mathrm{Var}(\boldsymbol{b}'\mathcal{X}) = \boldsymbol{b}'\boldsymbol{\Sigma}_{\mathcal{X}\mathcal{X}}\boldsymbol{b}$$

which is the positive, semi-definite condition. □

2.4 Mean and Variance of Quadratic Forms

Theorem 6. *Let $\mathcal{X}$ be a k-dimensional random vector and $\boldsymbol{A}$ be a constant $k \times k$ symmetric matrix. If $E(\mathcal{X}) = \boldsymbol{\mu}$ and $\mathrm{Var}(\mathcal{X}) = \boldsymbol{\Sigma}$, then*

$$E(\mathcal{X}'\boldsymbol{A}\mathcal{X}) = \mathrm{tr}(\boldsymbol{A}\boldsymbol{\Sigma}) + \boldsymbol{\mu}'\boldsymbol{A}\boldsymbol{\mu}$$

Proof.

$$\begin{aligned} E(\mathcal{X}'\boldsymbol{A}\mathcal{X}) &= \mathrm{tr}(E(\mathcal{X}'\boldsymbol{A}\mathcal{X})) \\ &= E[\mathrm{tr}(\mathcal{X}'\boldsymbol{A}\mathcal{X})] \\ &= E[\mathrm{tr}(\boldsymbol{A}\mathcal{X}\mathcal{X}')] \\ &= \mathrm{tr}(\boldsymbol{A}E[\mathcal{X}\mathcal{X}']) \\ &= tr(\boldsymbol{A}(\mathrm{Cov}(\mathcal{X}) + \boldsymbol{\mu}\boldsymbol{\mu}')) \\ &= tr(\boldsymbol{A}\boldsymbol{\Sigma}_{\mathcal{X}\mathcal{X}}) + \mathrm{tr}(\boldsymbol{A}\boldsymbol{\mu}\boldsymbol{\mu}') \\ &= tr(\boldsymbol{A}\boldsymbol{\Sigma}_{\mathcal{X}\mathcal{X}}) + \boldsymbol{\mu}'\boldsymbol{A}\boldsymbol{\mu} \end{aligned}$$

□

2.5 Distribution of Quadratic Forms in Normal Random Variables

Definition 4 (Non-Central χ^2)**.** *If $\mathcal{X}$ is a (scalar) normal random variable with $E(\mathcal{X}) = \mu$ and $\mathrm{Var}(\mathcal{X}) = 1$, then the random variable $\mathcal{V} = \mathcal{X}^2$ is distributed as $\chi^2_1(\lambda^2)$, which is called the noncentral χ^2 distribution with 1 degree of freedom and non-centrality parameter $\lambda^2 = \mu^2$. The mean and variance of $\mathcal{V}$ are*

$$E[\mathcal{V}] = 1 + \lambda^2 \text{ and } \mathrm{Var}[\mathcal{V}] = 2 + 4\lambda^2$$

As described in the previous chapter, we are particularly interested in random n-vectors, $\boldsymbol{Y}$, that have a *spherical normal distribution.*

Theorem 7. *Let $\mathcal{Y} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2 \boldsymbol{I}_n)$ be an n-vector with a spherical normal distribution and $\boldsymbol{A}$ be an $n \times n$ symmetric matrix. Then the ratio $\mathcal{Y}'\boldsymbol{A}\mathcal{Y}/\sigma^2$ will have a $\chi^2_r(\lambda^2)$ distribution with $\lambda^2 = \boldsymbol{\mu}'\boldsymbol{A}\boldsymbol{\mu}/\sigma^2$ if and only if $\boldsymbol{A}$ is idempotent with rank$(\boldsymbol{A}) = r$*

Proof. Suppose that $\boldsymbol{A}$ is idempotent (which, in combination with being symmetric, means that it is a projection matrix) and has $rank(\boldsymbol{A}) = r$. Its eigendecomposition, $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}'$, is such that $\boldsymbol{V}$ is orthogonal and $\boldsymbol{\Lambda}$ is $n \times n$ diagonal with exactly $r = \text{rank}(\boldsymbol{A})$ ones and $n - r$ zeros on the diagonal. Without loss of generality we can (and do) arrange the eigenvalues in decreasing order so that $\lambda_j = 1$, $j = 1, \dots, r$ and $\lambda_j = 0$, $j = r+1, \dots, n$ Let $\mathcal{X} = \boldsymbol{V}'\mathcal{Y}$

$$\begin{aligned}\frac{\mathcal{Y}'\boldsymbol{A}\mathcal{Y}}{\sigma^2} &= \frac{\mathcal{Y}'\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}'\mathcal{Y}}{\sigma^2} \\ &= \frac{\mathcal{X}'\boldsymbol{\Lambda}\mathcal{X}}{\sigma^2} \\ &= \sum_{j=1}^{n} \lambda_j \frac{\mathcal{X}_j^2}{\sigma^2} \\ &= \sum_{j=1}^{r} \frac{\mathcal{X}_j^2}{\sigma^2}\end{aligned}$$

(Notice that the last sum is to $j = r$, not $j = n$.) However, $\frac{\mathcal{X}_j}{\sigma} \sim \mathcal{N}(\boldsymbol{v}_j'\boldsymbol{\mu}/\sigma, 1)$ so $\frac{\mathcal{X}_j^2}{\sigma^2} \sim \chi^2_1((\boldsymbol{v}_j'\boldsymbol{\mu}/\sigma)^2)$. Therefore

$$\sum_{j=1}^{r} \frac{\mathcal{X}_j^2}{\sigma^2} \sim \chi^2_{(r)}(\lambda^2) \text{ where } \lambda^2 = \frac{\boldsymbol{\mu}'\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}'\boldsymbol{\mu}}{\sigma^2} = \frac{\boldsymbol{\mu}'\boldsymbol{A}\boldsymbol{\mu}}{\sigma^2}$$

□

Corollary 5. *For $\boldsymbol{A}$ a projection of rank r, $(\mathcal{Y}'\boldsymbol{A}\mathcal{Y})/\sigma^2$ has a central χ^2 distribution if and only if $\boldsymbol{A}\boldsymbol{\mu} = \boldsymbol{0}$*

Proof. The χ^2_r distribution will be central if and only if

$$0 = \boldsymbol{\mu}'\boldsymbol{A}\boldsymbol{\mu} = \boldsymbol{\mu}'\boldsymbol{A}\boldsymbol{A}\boldsymbol{\mu} = \boldsymbol{\mu}'\boldsymbol{A}'\boldsymbol{A}\boldsymbol{\mu} = \|\boldsymbol{A}\boldsymbol{\mu}\|^2$$

□

Corollary 6. *In the full-rank Gaussian linear model, $\boldsymbol{Y} \sim \mathcal{N}(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2\boldsymbol{I}_n)$, the residual sum of squares, $\|\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}\|^2$ has a central $\sigma^2\chi^2_{n-r}$ distribution.*

Proof. In the full rank model with the QR decomposition of $\boldsymbol{X}$ given by

$$\boldsymbol{X} = \begin{bmatrix}\boldsymbol{Q}_1 & \boldsymbol{Q}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{R}\\ \mathbf{0}\end{bmatrix}$$

and $\boldsymbol{R}$ invertible, the fitted values are $\boldsymbol{Q}_1\boldsymbol{Q}_1'\mathcal{Y}$ and the residuals are $\boldsymbol{Q}_2\boldsymbol{Q}_2\boldsymbol{y}$ so the residual sum of squares is the quadratic form $\mathcal{Y}'\boldsymbol{Q}_2\boldsymbol{Q}_2'\mathcal{Y}$. The matrix defining the quadratic form, $\boldsymbol{Q}_2\boldsymbol{Q}_2'$, is a projection matrix. It is obviously symmetric and it is idempotent because $\boldsymbol{Q}_2\boldsymbol{Q}_2'\boldsymbol{Q}_2\boldsymbol{Q}_2' = \boldsymbol{Q}_2\boldsymbol{Q}_2'$. As

$$\boldsymbol{Q}_2'\boldsymbol{\mu} = \boldsymbol{Q}_2'\boldsymbol{X}\boldsymbol{\beta}_0 = \boldsymbol{Q}_2'\boldsymbol{Q}_1\boldsymbol{R}\boldsymbol{\beta}_0 = \underbrace{\mathbf{0}}_{(n-p)\times n}\boldsymbol{R}\boldsymbol{\beta}_0 = \underbrace{\mathbf{0}}_{(n-p)\times p}\boldsymbol{\beta}_0 = \mathbf{0}_{n-p}$$

the ratio

$$\frac{\mathcal{Y}'\boldsymbol{Q}_2\boldsymbol{Q}_2'\mathcal{Y}}{\sigma^2} \sim \chi^2_{n-p}$$

and the RSS has a central $\sigma^2\chi^2_{n-p}$ distribution. □

R Exercises: Let's check some of these results by simulation. First we claim that if $\mathcal{X} \sim \mathcal{N}(\mu, 1)$ then $\mathcal{X}^2 \sim \chi^2(\lambda^2)$ where $\lambda^2 = \mu^2$. First simulate from a standard normal distribution

```
> set.seed(1234)                       # reproducible "random" values
> X <- rnorm(100000)                   # standard normal values
> zapsmall(summary(V <- X^2))          # a very skew distribution

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.1026  0.4521  0.9989  1.3190 20.3300

> var(V)

[1] 1.992403
```

The mean and variance of the simulated values agree quite well with the theoretical values of 1 and 2, respectively.

To check the form of the distribution we could plot an empirical density function but this distribution has its maximum density at 0 and is zero to the left of 0 so an empirical density is a poor indication of the actual shape of the density. Instead, in Fig. 2.1, we present the quantile-quantile plot for this sample versus the (theoretical) quantiles of the χ^2_1 distribution.

Now simulate a non-central χ^2 with non-centrality parameter $\lambda^2 = 4$

```
> V1 <- rnorm(100000, mean=2)^2
> zapsmall(summary(V1))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   1.773   3.994   5.003   7.144  39.050

> var(V1)
```

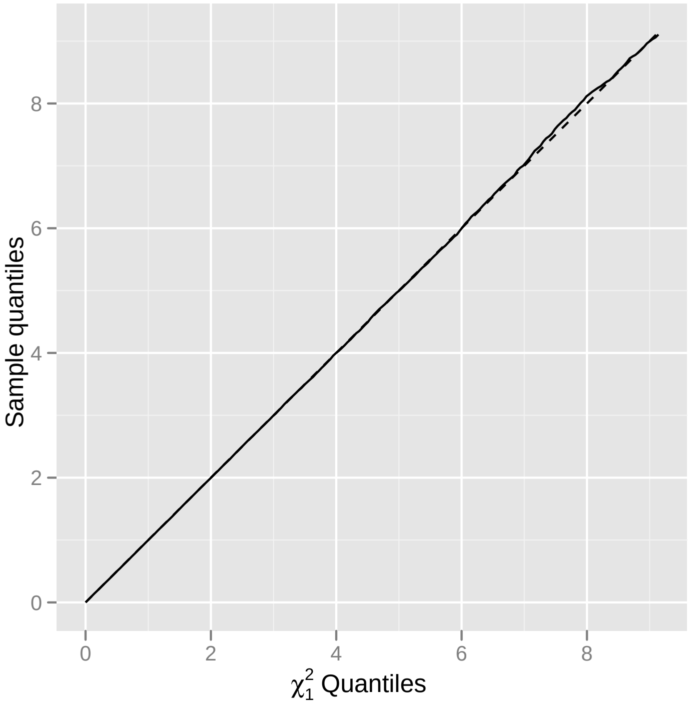

Figure 2.1: A quantile-quantile plot of the squares of simulated $\mathcal{N}(0,1)$ random variables versus the quantiles of the χ^2_1 distribution. The dashed line is a reference line through the origin with a slope of 1.

```
[1] 17.95924
```

The sample mean is close to the theoretical value of $5 = 1 + \lambda^2$ and the sample variance is close to the theoretical value of $2 + 4\lambda^2$ although perhaps not as close as one would hope in a sample of size 100,000.

A quantile-quantile plot versus the non-central distribution, $\chi^2_1(4)$, (Fig. 2.2) and versus the central distribution, χ^2_1, shows that the sample does follow the claimed distribution $\chi^2_1(4)$ and is stochastically larger than the χ^2_1 distribution.

More interesting, perhaps is the distribution of the residual sum of squares from a regression model. We simulate from our previously fitted model `lm1`

```
> lm1 <- lm(optden ~ carb, Formaldehyde)
> str(Ymat <- data.matrix(unname(simulate(lm1, 10000))))

 num [1:6, 1:10000] 0.088 0.258 0.444 0.521 0.619 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:6] "1" "2" "3" "4" ...
  ..$ : NULL
```

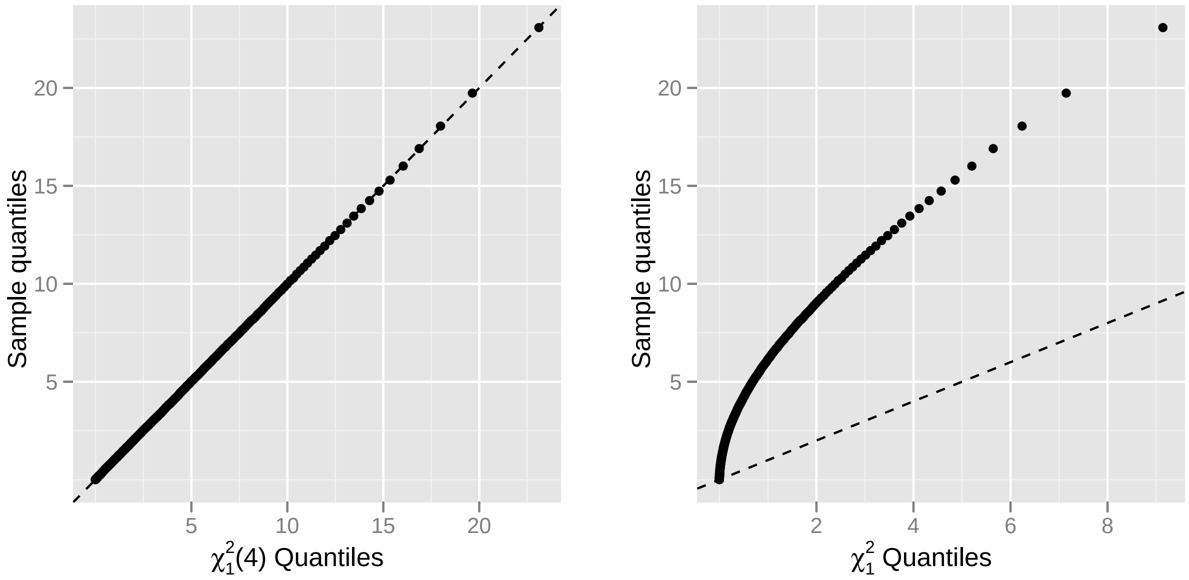

Figure 2.2: Quantile-quantile plots of a sample of squares of $\mathcal{N}(2, 1)$ random variables versus the quantiles of a $\chi^2_1(4)$ non-central distribution (left panel) and a χ^2_1 central distribution (right panel)

```
> str(RSS <- deviance(fits <- lm(Ymat ~ carb, Formaldehyde)))

 num [1:10000] 0.000104 0.000547 0.00055 0.000429 0.000228 ...

> fits[["df.residual"]]

[1] 4
```

Here the `Ymat` matrix is 10,000 simulated response vectors from model `lm1` using the estimated parameters as the true values of $\boldsymbol{\beta}$ and σ^2. Notice that we can fit the model to **all** 10,000 response vectors in a single call to the `lm()` function.

The `deviance()` function applied to a model fit by `lm()` returns the residual sum of square, which is not technically the deviance but is often the quantity of interest.

These simulated residual sums of squares should have a $\sigma^2\chi^2_4$ distribution where σ^2 is the residual sum of squares in model `lm1` divided by 4.

```
> (sigsq <- deviance(lm1)/4)

[1] 7.48e-05

> summary(RSS)

     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
6.997e-07 1.461e-04 2.537e-04 3.026e-04 4.114e-04 1.675e-03
```

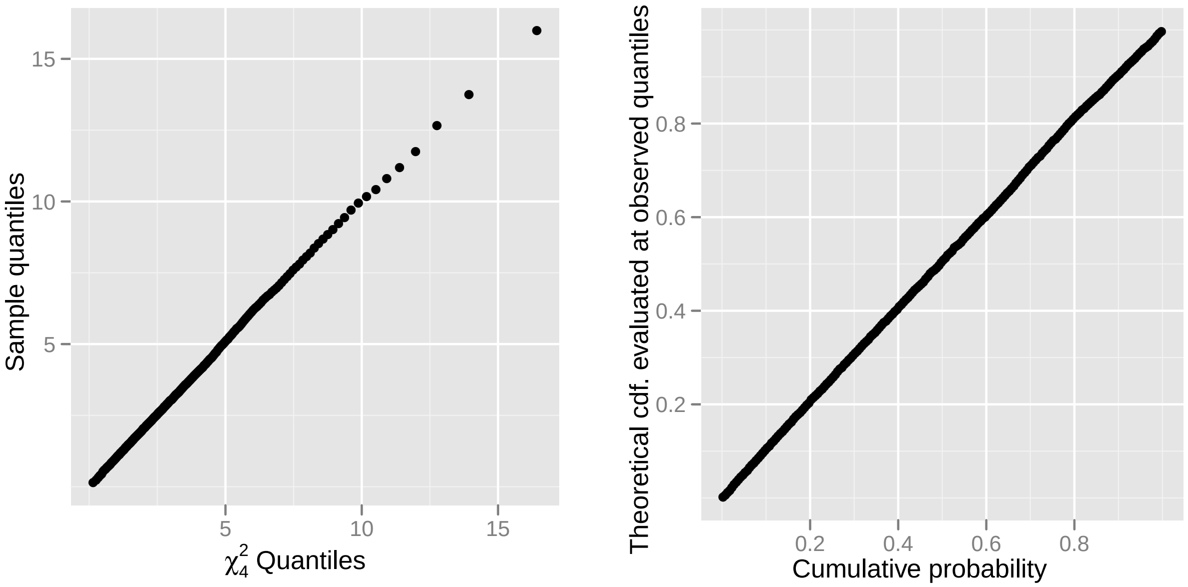

Figure 2.3: Quantile-quantile plot of the scaled residual sum of squares, `RSSsq`, from simulated responses versus the quantiles of a χ^2_4 distribution (left panel) and the corresponding probability-probability plot on the right panel.

We expect a mean of $4\sigma^2$ and a variance of $2 \cdot 4(\sigma^2)^2$. It is easier to see this if we divide these values by σ^2

```
> summary(RSSsc <- RSS/sigsq)

     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
 0.009354  1.953000  3.392000  4.045000  5.500000 22.390000

> var(RSSsc)

[1] 8.000057
```

A quantile-quantile plot with respect to the χ^2_4 distribution (Fig. 2.3) shows very good agreement between the empirical and theoretical quantiles. Also shown in Fig. 2.3 is the probability-probability plot. Instead of plotting the sample quantiles versus the theoretical quantiles we take equally spaced values on the probability scale (function `ppoints()`), evaluate the sample quantiles and then apply the theoretical cdf to the empirical quantiles. This should also produce a straight line. It has the advantage that the points are equally spaced on the x-axis.

We could also plot the empirical density of these simulated values and overlay it with the theoretical density (Fig. 2.4).

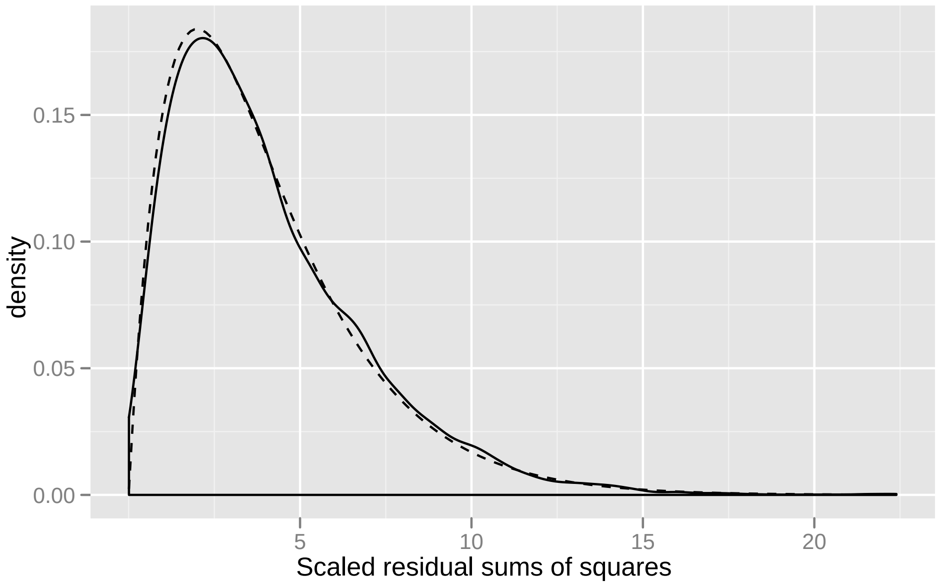

Figure 2.4: Empirical density plot of the scaled residual sums of squares, `RSSsq`, from simulated responses. The overlaid dashed line is the density of a χ^2_4 random variable. The peak of the empirical density gets shifted a bit to the right because of the way the empirical density if calculated. It uses a symmetric kernel which is not a good choice for a skewed density like this.

Chapter 3

Properties of coefficient estimates

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

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

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

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

3.1 Geometric Properties

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

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

The dimension of $\mathrm{col}(\boldsymbol{X})$ is $k = \mathrm{rank}(\boldsymbol{X})$ 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 $\mathrm{col}(\boldsymbol{X})$.

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 $\mathrm{col}(\boldsymbol{X})$

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

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

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

The diagonal matrix, $\boldsymbol{D}$, in the singular value decomposition (sect. 1.4.3, p. 17), $\boldsymbol{X} = \boldsymbol{U}_1\boldsymbol{DV}'$, 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 $\mathrm{col}(\boldsymbol{X})$.

The residual at the parameter estimates, $\widehat{\boldsymbol{e}} = \boldsymbol{y} - \widehat{\boldsymbol{y}}$ is orthogonal to $\mathrm{col}(\boldsymbol{X})$. 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 $\mathrm{col}(\boldsymbol{X})$.

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

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, $\arg\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \boldsymbol{X\beta}\|^2$, you use orthogonal projection of $\boldsymbol{y}$ onto $\mathrm{col}(\boldsymbol{X})$ which implies that the residual is orthogonal to $\mathrm{col}(\boldsymbol{X})$.

Often this relationship is characterized as the *normal equations*. The residual will be orthogonal to $\mathrm{col}(\boldsymbol{X})$ 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

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

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

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

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

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 $1/d_{i,i}$, $i = 1, \dots, k$ 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

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

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)
```

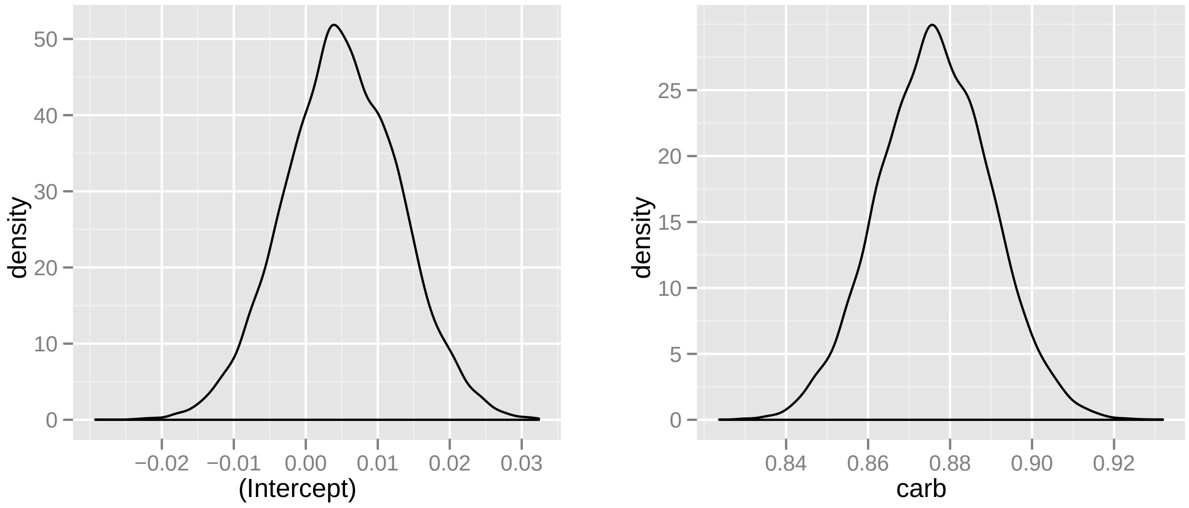

Figure 3.1: Empirical density plots of coefficient estimates from data simulated according to the estimated parameters in model `lm1`

```
              (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.

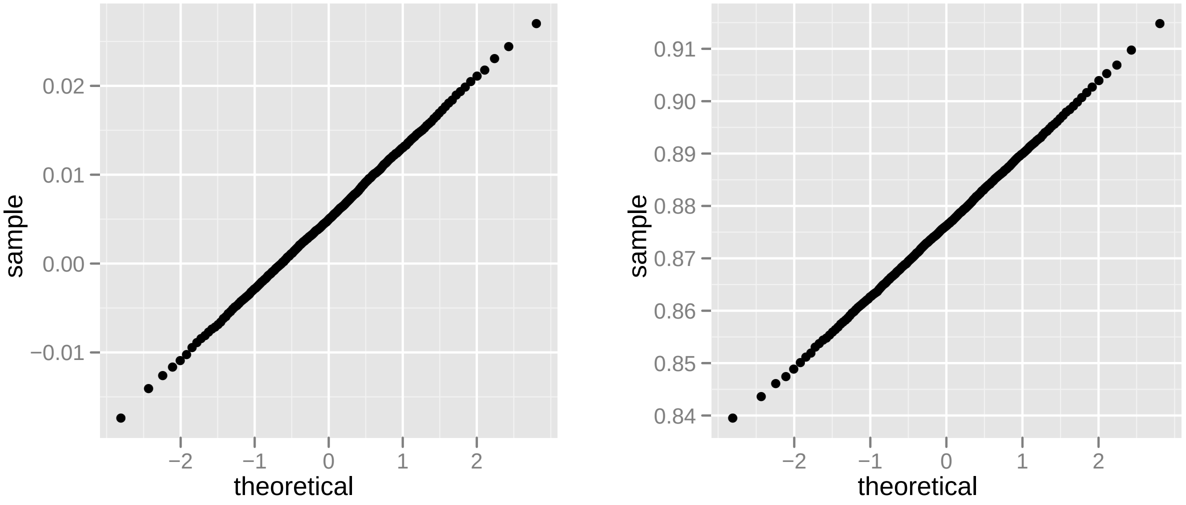

Figure 3.2: Normal quantile-quantile plots of coefficient estimates from responses simulated according to the estimated parameters in model `lm1`.

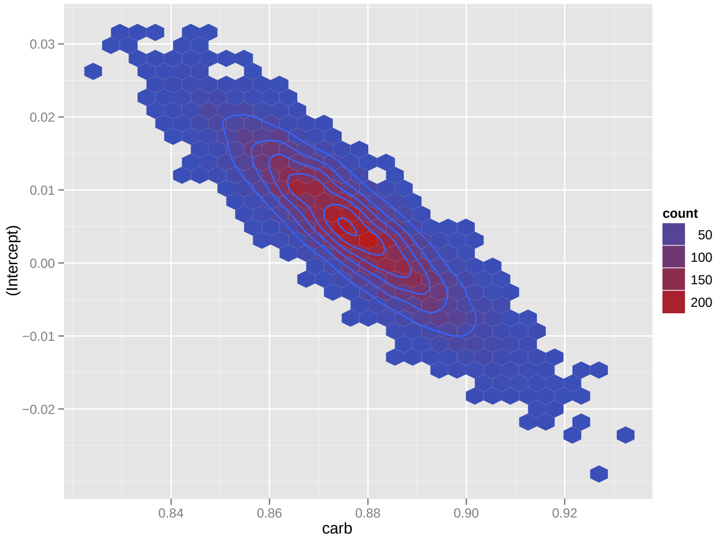

Figure 3.3: Normal quantile-quantile plots of coefficient estimates from responses simulated according to the estimated parameters in model `lm1`.

Chapter 4

The Gauss-Markov Theorem

In Chap. 3 we showed that the least squares estimator, $\widehat{\boldsymbol{\beta}}_{LSE}$, in a Gaussian linear model has is *unbiased*, meaning that $E[\widehat{\boldsymbol{\beta}}_{LSE}] = \boldsymbol{\beta}$, and that its variance-covariance matrix is

$$\operatorname{Var} \widehat{\boldsymbol{\beta}}_{LSE} = \sigma^2 \left(\boldsymbol{X}'\boldsymbol{X}\right)^{-1} = \sigma^2 \boldsymbol{R}^{-1}(\boldsymbol{R}^{-1})'.$$

The Gauss-Markov theorem says that this variance-covariance (or *dispersion*) is the best that we can do when we restrict ourselved to *linear unbiased estimators*, which means estimators that are linear functions of $\mathcal{Y}$ and are unbiased.

To make these definitions more formal:

Definition 5 (Minimum Dispersion). *Let $\mathcal{T} = (\mathcal{T}_1, \ldots, \mathcal{T}_p)'$ be an estimator of $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)'$. The dispersion of $\mathcal{T}$ is $\boldsymbol{D}(\mathcal{T}) = E[(\mathcal{T} - \boldsymbol{\theta})(\mathcal{T} - \boldsymbol{\theta})']$. If $\mathcal{T}$ is unbiased then its dispersion is simply its variance-covariance matrix, $\boldsymbol{D}(\mathcal{T}) = \operatorname{Var}(\mathcal{T})$. $\mathcal{T}$ is minimum dispersion unbiased estimator of $\boldsymbol{\theta}$ if $\boldsymbol{D}(\tilde{\mathcal{T}}) - \boldsymbol{D}(\mathcal{T})$ is positive semidefinite for any unbiased estimator $\tilde{\mathcal{T}}$. That is*

$$\boldsymbol{a}'[\boldsymbol{D}(\tilde{\mathcal{T}}) - \boldsymbol{D}(\mathcal{T})]\boldsymbol{a} \geq 0 \quad \forall\, \boldsymbol{a} \in \mathbb{R}^p$$

Because the dispersion matrices of unbiased estimators are the variance-covariance matrices, this condition is equivalent to

$$\boldsymbol{a}' \operatorname{Var}(\tilde{\mathcal{T}})\boldsymbol{a} - \boldsymbol{a}' \operatorname{Var}(\mathcal{T})\boldsymbol{a} \geq 0 \Rightarrow \operatorname{Var}(\boldsymbol{a}'\tilde{\mathcal{T}}) - \operatorname{Var}(\boldsymbol{a}'\mathcal{T}) \geq 0$$

Theorem 8 (Gauss-Markov). *In the full-rank case (i.e.* $\operatorname{rank}(\boldsymbol{X}) = p$*) the minimum dispersion linear unbiased estimator of $\boldsymbol{\beta}$ is $\widehat{\boldsymbol{\beta}}_{LSE}$ with dispersion matrix $\sigma^2(\boldsymbol{X}'\boldsymbol{X})^{-1}$. It is also called the* best linear unbiased estimator *or BLUE of $\boldsymbol{\beta}$.*

Proof. Any linear estimator of $\boldsymbol{\beta}$ can be written as $\boldsymbol{A}\mathcal{Y}$ for some $p \times n$ matrix $\boldsymbol{A}$. (That's what it means to be a linear estimator.) To be an unbiased linear estimator we must have

$$\boldsymbol{\beta} = E[\boldsymbol{A}\mathcal{Y}] = \boldsymbol{A}\, E[\mathcal{Y}] = \boldsymbol{A}\, \boldsymbol{X}\boldsymbol{\beta} \quad \forall\, \boldsymbol{\beta} \in \mathbb{R}^p \;\Rightarrow\; \boldsymbol{A}\boldsymbol{X} = \boldsymbol{I}_p$$

The variance-covariance matrix such a linear unbiased estimator, $\boldsymbol{A}\mathcal{Y}$, is

$$\operatorname{Var}(\boldsymbol{A}\mathcal{Y}) = \boldsymbol{A} \operatorname{Var}(\mathcal{Y})\boldsymbol{A}' = \boldsymbol{A}\sigma^2\boldsymbol{I}_n\boldsymbol{A}' = \sigma^2\boldsymbol{A}\boldsymbol{A}'.$$

Now we must show that

$$\mathrm{Var}(\boldsymbol{a}'\boldsymbol{A}\mathcal{Y}) - \mathrm{Var}(\boldsymbol{a}'\widehat{\boldsymbol{\beta}}_{LSE}) = \sigma^2\boldsymbol{a}'\left(\boldsymbol{A}\boldsymbol{A}' - (\boldsymbol{X}'\boldsymbol{X})^{-1}\right)\boldsymbol{a} \geq 0,\ \forall\ \boldsymbol{a} \in \mathbb{R}^p.$$

In other words, the symmetric matrix, $\left(\boldsymbol{A}\boldsymbol{A}' - (\boldsymbol{X}'\boldsymbol{X})^{-1}\right)$, must be positive semi-definite. Consider

$$\begin{aligned}\boldsymbol{A}\boldsymbol{A}' =&[\boldsymbol{A} - (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}' + (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'][\boldsymbol{A} - (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}' + (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}']' \\ =&[\boldsymbol{A} - (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'][\boldsymbol{A} - (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}']' + [\boldsymbol{A} - (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'][(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}]' + \\ &(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}[\boldsymbol{A} - (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}']' + [(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'][\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}] \\ =&[\boldsymbol{A} - (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'][\boldsymbol{A} - (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}']' + (\boldsymbol{X}'\boldsymbol{X})^{-1},\end{aligned}$$

showing that $\boldsymbol{A}\boldsymbol{A}' - (\boldsymbol{X}'\boldsymbol{X})^{-1}$ is the positive semi-definite matrix $[\boldsymbol{A} - (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'][\boldsymbol{A} - (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}']'$. Therefore $\widehat{\boldsymbol{\beta}}_{LSE}$ is the BLUE for $\boldsymbol{\beta}$. □

Corollary 7. *If* $\mathrm{rank}(\boldsymbol{X}) = p < n$, *the best linear unbiased estimator of* $\boldsymbol{a}'\boldsymbol{\beta}$ *is* $\boldsymbol{a}'\widehat{\boldsymbol{\beta}}_{LSE}$.

To extend the Gauss-Markov theorem to the rank-deficient case we must define

Definition 6 (Estimable linear function)**.** *An estimable linear function of the parameters* $\boldsymbol{\beta}$ *in the linear model,* $\mathcal{Y} \sim \mathcal{N}(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2\boldsymbol{I}_n)$, *is any function of the form* $\boldsymbol{l}'\boldsymbol{\beta}$ *where* $\boldsymbol{l}$ *is in the row span of* $\boldsymbol{X}$. *That is,* $\boldsymbol{l}'\boldsymbol{\beta}$ *is estimable if and only if there exists* $\boldsymbol{c} \in \mathbb{R}^n$ *such that* $\boldsymbol{l} = \boldsymbol{X}'\boldsymbol{c}$.

The coefficients of the estimable functions form a $\mathrm{rank}(\boldsymbol{X}) = k$-dimensional linear subspace of $\mathbb{R}^p$. In the full-rank this subspace is all of $\mathbb{R}^p$ so any linear combination $\boldsymbol{l}'\boldsymbol{\beta}$ is estimable.

In the rank-deficient case (i.e. $\mathrm{rank}(\boldsymbol{X}) = k < p$), consider the singular value decomposition $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}'$ with $\boldsymbol{D}$ a diagonal matrix having non-negative, non-increasing diagonal elements, the first k of which are positive and the last $p-k$ are zero. Let $\boldsymbol{U}_k$ be the first k columns of $\boldsymbol{U}$, $\boldsymbol{D}_k$ be the first k rows and k columns of $\boldsymbol{D}$, and $\boldsymbol{V}_k$ be the first k columns of $\boldsymbol{V}$. The coefficients $\boldsymbol{l}$ for an estimable linear function must lie in the column span of $\boldsymbol{V}_k$ because

$$\boldsymbol{l} = \boldsymbol{X}'\boldsymbol{c} = \boldsymbol{V}_k\underbrace{\boldsymbol{D}_k\boldsymbol{U}_k'\boldsymbol{c}}_{\boldsymbol{a}} = \boldsymbol{V}_k\boldsymbol{a}$$

We will write the $p \times (p-k)$ matrix formed by the last $p-k$ columns of $\boldsymbol{V}$ as $\boldsymbol{V}_{p-k}$ so that

$$\boldsymbol{\beta} = \boldsymbol{V}\boldsymbol{V}'\boldsymbol{\beta} = \begin{bmatrix}\boldsymbol{V}_k & \boldsymbol{V}_{(p-k)}\end{bmatrix}\begin{bmatrix}\boldsymbol{V}_k' \\ \boldsymbol{V}_{p-k}'\end{bmatrix}\boldsymbol{\beta} = \boldsymbol{V}_k\boldsymbol{\gamma} + \boldsymbol{V}_{p-k}\boldsymbol{\delta}$$

where $\boldsymbol{\gamma} = \boldsymbol{V}_k'\boldsymbol{\beta}$ and $\boldsymbol{\delta} = \boldsymbol{V}_{p-k}'\boldsymbol{\beta}$ are the estimable and inestimable parts of the parameter vector in the $\boldsymbol{V}$ basis.

Now any estimable function is of the form

$$\boldsymbol{l}'\boldsymbol{\beta} = \boldsymbol{a}'\boldsymbol{V}_k'\boldsymbol{\beta} = \boldsymbol{a}'\boldsymbol{\gamma} + \boldsymbol{0} = \boldsymbol{a}'\boldsymbol{\gamma},$$

where $\boldsymbol{\gamma}$ is the parameter in the full-rank model $\mathcal{Y} \sim \mathcal{N}(\boldsymbol{D}_k\boldsymbol{U}_k\boldsymbol{\gamma}, \sigma^2\boldsymbol{I}_n)$.

So anything we say about estimable functions of $\boldsymbol{\beta}$ can be transformed into a statement about $\boldsymbol{\gamma}$ in the full rank model and anything we say about the fitted values, $\boldsymbol{X}\boldsymbol{\beta}$, or the residuals can be expressed in terms of the full-rank $\boldsymbol{D}_k\boldsymbol{U}_k\boldsymbol{\gamma}$. In particular, the hat matrix, $\boldsymbol{H} = \boldsymbol{U}_k\boldsymbol{U}_k'$, and has $\mathrm{rank}(\boldsymbol{H}) = k$ and the projection into the orthogonal (residual) space is $\boldsymbol{I}_n - \boldsymbol{H}$.

Corollary 8 (Gauss-Markov extension to rank-deficient cases)**.** *$\boldsymbol{l}'\widehat{\boldsymbol{\beta}}_{LSE} = \boldsymbol{a}'\widehat{\boldsymbol{\gamma}}_{LSE}$ is the BLUE for any estimable linear function, $\boldsymbol{l}'\boldsymbol{\beta}$, of $\boldsymbol{\beta}$.*

Proof. By the Gauss-Markov theorem $\widehat{\boldsymbol{\gamma}}_{LSE}$ is the BLUE for $\boldsymbol{\gamma}$ and $\boldsymbol{l}'\boldsymbol{\beta} = \boldsymbol{a}'\boldsymbol{\gamma}$ is a linear function of $\boldsymbol{\gamma}$. □

Theorem 9. *Suppose that $k = rank(\boldsymbol{X}) \leq p$. Then an unbiased estimator of σ^2 is*

$$S^2 = \frac{\|\mathcal{Y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}\|^2}{n-k} = \frac{\|\hat{\boldsymbol{\epsilon}}\|^2}{n-k} = \frac{\sum_{i=1}^n \hat{\epsilon}_i^2}{n-k}.$$

Proof. The simple proof is to observe that this estimator is the unbiased estimator of σ^2 for the full-rank version of the model, $\mathcal{Y} \sim \mathcal{N}(\boldsymbol{D}_k\boldsymbol{U}_k\boldsymbol{\gamma}, \sigma^2\boldsymbol{I}_n)$. □