2.1 Multiple Linear Regression Model

Multiple Linear Regression (MLR) is a natural extension of Simple Linear Regression by adding more predictors. From a practical point of view, this is necessary when the variation left in a SLR model (i.e. RSS) is quite high.

We will start our discussion about MLR with the `Birthweight` example:

Birthweight Example

The `birthweight.csv` data set contains data from a study on the birth weight of **42** babies. The variables in the data set are the following:

- `ID` : Unique Identification number of a baby
- `Length` : Length of the baby at time of birth in cm (X_1)
- `Birthweight` : Weight of the baby at time of birth in kg (Y)
- `Headcirc` : Head circumference of the baby at time of birth in cm (X_2)
- `Gestation` : Completed weeks of gestation (X_3)
- `smoker` : Mother is/is not a smoker {0: No, 1: Yes} (X_4)
- `mage` : Mother's age at time of birth (X_5)
- `mnocig` : Mother's number of cigarettes smoked per day (X_6)
- `mheight` : Mother's height in cm (X_7)
- `mppwt` : Mother's pre-pregnancy weight (X_8)
- `fage` : Father's age at time of birth (X_9)
- `fedyrs` : Father's years of education (X_{10})
- `fnocig` : Father's number of cigarettes smoked per day (X_{11})
- `fheight` : Father's height (X_{12})
- `lowbwt` : Low birthweight {0: No, 1:Yes} (X_{13})
- `mage35` : Mother's Age $\geq$ 35 or not {0: No, 1: Yes}

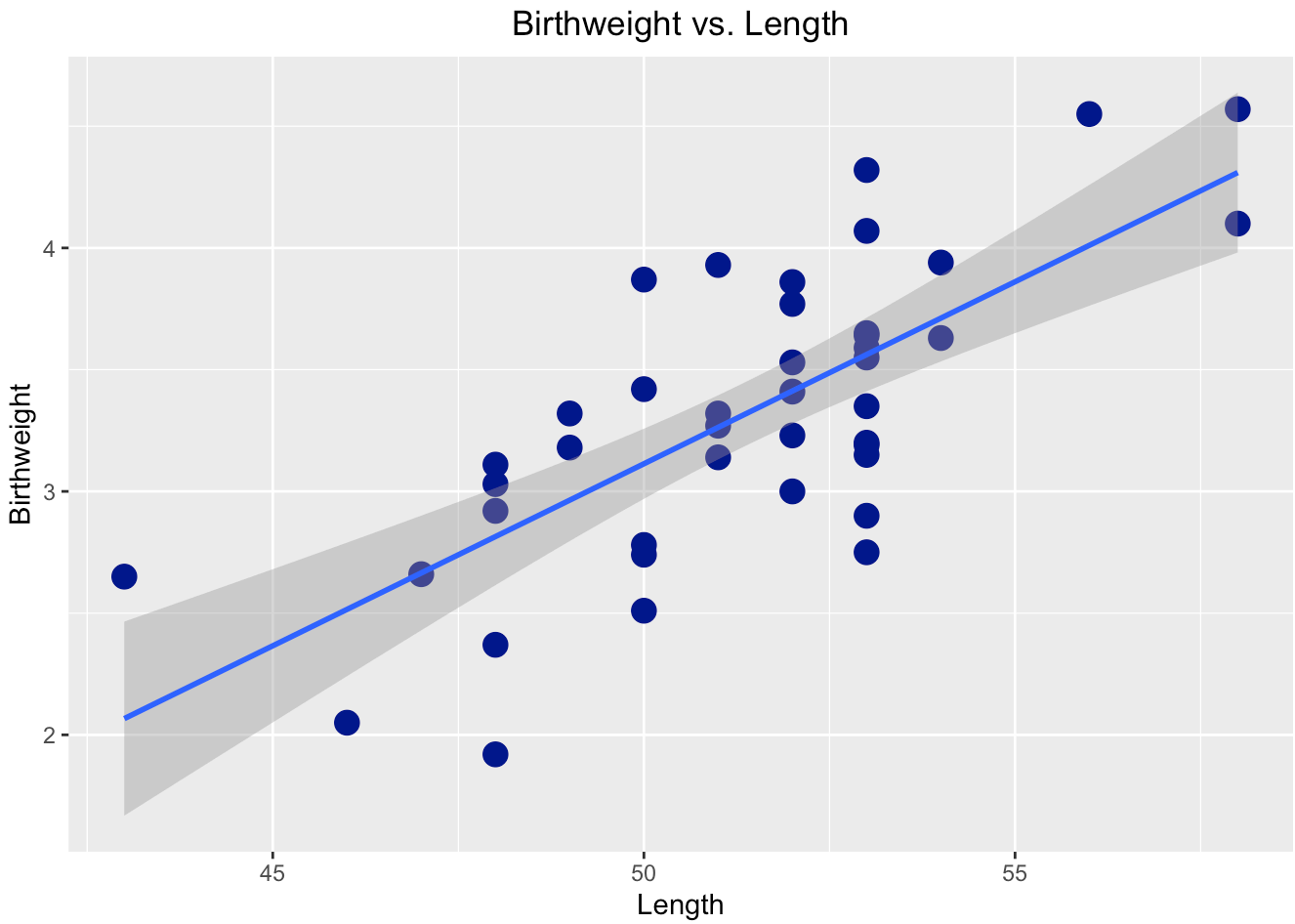
```
birthweight <- read.csv("data/ch2/Birthweight.csv", header=TRUE)
```

Our goal in this example is to estimate the **mean** `Birthweight`. Based on the literature, we know that the weight of a baby is related to its length. So, let's start by running a SLR between `Birthweight` (the response) and `Length` a single predictor.

```
birthweight.slr = lm(Birthweight ~ Length, data=birthweight)
summary(birthweight.slr)
```

```
##
## Call:
## lm(formula = Birthweight ~ Length, data = birthweight)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.89446 -0.35492  0.01746  0.28674  0.75794
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.36244    1.14858  -3.798 0.000486 ***
## Length       0.14952    0.02234   6.693 5.03e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4199 on 40 degrees of freedom
## Multiple R-squared:  0.5283, Adjusted R-squared:  0.5165
## F-statistic:  44.8 on 1 and 40 DF,  p-value: 5.029e-08
```

We also plot the data with the fitted regression line on top:



Looking both at the regression output and the scatterplot, we observe that there is still a lot of variation in the data that is not explained by `Length` alone. This means that other factors/variables are (possibly) responsible for the variation that has not been explained by `Length` .

In many situations, like the example above, a **single** predictor variable in the model provides an inadequate description, since a **number of key variables** affect the response variable in important and distinctive ways. In addition, in situations like this, we frequently find that predictions of the response variable based on a model containing only a single predictor are too imprecise to be useful. A more complex model, containing additional predictor variables, typically is more helpful in providing sufficiently precise predictions of the response.

2.1.1 MLR Formulation

We consider a model with more than one predictor. Suppose x_1, x_2, ..., x_p be p **predictors** of a *response* y. In this case, the data will be of the form

$$\begin{array}{cccccc} y_1 & x_{11} & x_{12} & \cdots & x_{1p} \\ y_2 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_n & x_{n1} & x_{n2} & \cdots & x_{np} \end{array}$$

and the model can be written as:

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i, \qquad i = 1, \ldots, n$$

where we denote $\mathbf{x_i} = (x_{i1,\ldots,x_ip})^T$, with $x_{i1} = 1$.

Here $(\beta_1, \beta_2, \ldots, \beta_p; \sigma^2)$ are *unknown true* **parameters**.

- β_1 is the *intercept*, which means that $x_{i1} = 1, i = 1, \ldots, n$.
- $\beta_2, \beta_3, \ldots, \beta_p$ are *partial slopes*.
- $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ are the *random errors*. As usual, we assume that they satisfy the same conditions as in the case of simple linear regression:
  - zero mean: $\mathbb{E}(\varepsilon_i) = 0$; i,
  - uncorrelated: $Cov(\varepsilon_i, \varepsilon_j) = 0,\ i \neq j$, and
  - homoscedastic: $Var(\varepsilon_i) = \sigma^2$ (i.e. does not depend on i).

In the `Birthweight` example, a snapshot of the data is shown below:

```
head(birthweight)
```

```
##     ID Length Birthweight Headcirc Gestation smoker mage mnocig mheight mppwt
## 1 1360     56        4.55       34        44      0   20      0     162    57
## 2 1016     53        4.32       36        40      0   19      0     171    62
## 3  462     58        4.10       39        41      0   35      0     172    58
## 4 1187     53        4.07       38        44      0   20      0     174    68
## 5  553     54        3.94       37        42      0   24      0     175    66
## 6 1636     51        3.93       38        38      0   29      0     165    61
##   fage fedyrs fnocig fheight lowbwt mage35
## 1   23     10     35     179      0      0
## 2   19     12      0     183      0      0
## 3   31     16     25     185      0      1
## 4   26     14     25     189      0      0
## 5   30     12      0     184      0      0
## 6   31     16      0     180      0      0
```

Here Column 3 (`Birthweight`) is the response and all other columns are predictors. Using the notation defined above, we have $n = 42$.

```
dim(birthweight)
```

```
## [1] 42 16
```

We are going to write our Multiple Linear Regression model in matrix formulation. First define the following vectors/matrices:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}$$

Using this notation the model equation can be written as:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

or

Matrix Representation of the MLR Model

$$\underset{\uparrow}{\mathbf{y}_{n\times 1}} = \underset{\uparrow}{\mathbf{X}_{n\times p}} \underset{\uparrow}{\beta_{p\times 1}} + \underset{\uparrow}{\varepsilon_{n\times 1}}$$

$$\begin{matrix} \text{Response} & & \text{Design Matrix} & \text{Coefficients} & & \text{Error Term} \end{matrix}$$

n: sample size

p: number of predictors or columns of $\mathbf{X}$

$\mathbf{X}$ is called the **design matrix**

By default the intercept is included in the model in which case the first column of $\mathbf{X}$ is a vector of 1's.

2.1.2 Least-Squares Estimation in MLR

We want to estimate the **vector** of β coefficients, i.e. obtain:

$$\hat{\beta} = \left(\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_p\right)^T$$

The LS estimator of β minimizes the *sum of squared residuals*:

$$RSS = ||y - \mathbf{X}\beta||^2 = (y - \mathbf{X}\beta)^T(y - \mathbf{X}\beta)$$

In order to minimize RSS, we take derivatives with respect to β's and set to zero (similar to what we did in SLR but in higher dimensions).

$$\frac{\partial RSS}{\partial \beta} = \mathbf{0}_{p\times 1} \Leftrightarrow$$
$$-2\ \mathbf{X}^T_{p\times n}(y - \mathbf{X}\beta)_{n\times 1} = \mathbf{0}_{p\times 1}$$

This leads to the so-called **Normal Equations**

$$\mathbf{X}^T(y - \mathbf{X}\beta) = \mathbf{0}$$

Solving the Normal Equations

$$(\mathbf{X}^T\mathbf{X})\ \beta = \mathbf{X}^T\ y$$

leads to the

Least Square Estimators in the MLR

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\ y$$

We assume that **the rank of $\mathbf{X}$ is $\mathbf{p}$**, i.e. no columns of $\mathbf{X}$ are a linear combinations of the other columns of $\mathbf{X}$. *Since $\mathbf{X}$ has rank p, the inverse of $(\mathbf{X}^T\mathbf{X})$ exists*.

Single Predictor Model in Matrix Format

Just to better understand the formula we derived in the MLR model for the β estimator, we do the calculations *"by hand"* in the case of a single predictor model

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i,\ n = 1, \ldots, n$$

If we re-write it in matrix format, we have:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

Use the formula $\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y$ to obtain the estimator from the previous lecture. We compute each term in the expression of $\hat{\beta}$:

$$\mathbf{X}^T\mathbf{X} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_n \end{pmatrix} \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} = \begin{pmatrix} n & n\bar{x} \\ n\bar{x} & \sum_i x_i^2 \end{pmatrix}$$

$$(\mathbf{X}^T\mathbf{X})^{-1} = \frac{1}{n\sum_i x_i^2 - (n\bar{x})^2} \begin{pmatrix} \sum_i x_i^2 & -n\bar{x} \\ -n\bar{x} & n \end{pmatrix}$$

$$\mathbf{X}^T y = \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_n \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} n\bar{y} \\ \sum_i x_i y_i \end{pmatrix}$$

Combining all the above, we obtain that

$$\begin{aligned} \hat{\beta} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y \\ &= \frac{1}{n\sum_i x_i^2 - (n\bar{x})^2} \begin{pmatrix} \sum_i x_i^2 & -n\bar{x} \\ -n\bar{x} & n \end{pmatrix} \begin{pmatrix} n\bar{y} \\ \sum_i x_i y_i \end{pmatrix} \end{aligned}$$

So, $\hat{\beta}_1$ is given by

$$\hat{\beta}_1 = \frac{-n^2\bar{x}\bar{y} + n\sum_i x_i y_i}{n\sum_i x_i^2 - (n\bar{x})^2} = \frac{\sum_i x_i y_i - n\bar{x}\bar{y}}{\sum_i x_i^2 - n\bar{x}^2}$$

and similarly we can recover the formula for $\hat{\beta}_0$.

Let's see how we can fit a MLR in R :

Birthweight Example

We start by fitting the **full model** with *all the available variables*, that is

$$Y_i = \beta_1 X_1 + \ldots + \beta_{14} X_{14} + \varepsilon_i$$

Here we assume that X_1 is a column of 1's and thus corresponds to the intercept. In R this:

```
# For convenience remove the index column.
# We also remove `mage35` because it is highly correlated with `mage`:
birthweight2 = birthweight[,c(-1, -16)]

# Run a regression with Birthweight as the
# response and everything else as a predictor:
birthweight.mlr1 = lm(Birthweight~., data=birthweight2)
summary(birthweight.mlr1)
```

```
##
## Call:
## lm(formula = Birthweight ~ ., data = birthweight2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.38656 -0.26722 -0.06068  0.18271  0.60295
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.4286417  2.1583202  -1.589  0.12339
## Length       0.0257860  0.0336226   0.767  0.44954
## Headcirc     0.0850933  0.0297843   2.857  0.00798 **
## Gestation    0.0916226  0.0322518   2.841  0.00829 **
## smoker      -0.2198237  0.1728531  -1.272  0.21393
## mage        -0.0158203  0.0191605  -0.826  0.41597
## mnocig       0.0002094  0.0070011   0.030  0.97635
## mheight      0.0056438  0.0143947   0.392  0.69797
## mppwt        0.0084338  0.0116016   0.727  0.47329
## fage         0.0046535  0.0167998   0.277  0.78382
## fedyrs       0.0016448  0.0312497   0.053  0.95840
## fnocig       0.0040531  0.0041251   0.983  0.33424
## fheight     -0.0122665  0.0097854  -1.254  0.22037
## lowbwt      -0.1751779  0.2346936  -0.746  0.46164
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3367 on 28 degrees of freedom
## Multiple R-squared:  0.7877, Adjusted R-squared:  0.6891
## F-statistic: 7.989 on 13 and 28 DF,  p-value: 2.432e-06
```

In the output, we can see the estimated β coefficients. Notice that here we have 13 predictors and 1 intercept which makes $\mathbf{p = 14}$.

2.1.3 Fitted Values & Residuals

We can compute the **fitted values** of y based on the model as follows:

$$\begin{aligned}\hat{y}_{n\times 1} &= \mathbf{X}\hat{\beta} \\ &= \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\, y \\ &= \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\, y := \mathbf{H}_{n\times n}y_{n\times 1}\end{aligned}$$

The Hat Matrix

We define

$$\mathbf{H}_{n\times n} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$$

to be the hat matrix, since it returns the "y-hat" values.

We also define the **residuals**

$$\begin{aligned}\mathbf{r}_{n\times 1} &= y - \hat{y} \\ &= y - \mathbf{X}\hat{\beta} \\ &= y - \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\, y \\ &= y - \mathbf{H}y \\ &= (\mathbf{I} - \mathbf{H})y\end{aligned}$$

The residuals $\mathbf{r}$ are used to estimate the *error variance*:

$$\hat{\sigma}^2 = \frac{1}{n-p}\sum_i r_i^2 = \frac{RSS}{n-p}$$

Birthweight Example

We can extract all this information using `R` :

```r
# To obtain the fitted values
birthweight.fitted1 = birthweight.mlr1$fitted.values
head(birthweight.fitted1)
```

```
##        1        2        3        4        5        6
## 4.088181 3.717055 4.050885 4.369037 4.027310 3.551145
```

```r
# To obtain the residuals
birthweight.resids1 = birthweight.mlr1$residuals
head(birthweight.resids1)
```

```
##           1           2           3           4           5           6
##  0.46181854  0.60294509  0.04911492 -0.29903736 -0.08730952  0.37885536
```

Properties of the Residuals

The LS estimator is the β vector that satisfies the **normal equations**, that is

$$\mathbf{X}^T(y - \hat{y}) = \mathbf{X}^T(y - \mathbf{X}\hat{\beta}) = \mathbf{0}$$

This implies the following properties for the residuals $r_{n\times 1}$:

- The cross-products between the residual vector r and *each column of* $\mathbf{X}$ are zero, i.e.

$$\begin{aligned}\mathbf{X}^T\, r &= \mathbf{X}^T(y - \mathbf{X}\hat{\beta}) \\ &= \mathbf{X}^T y - \mathbf{X}^T\mathbf{X}\hat{\beta} \\ &= \mathbf{X}^T y - (\mathbf{X}^T\mathbf{X})(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\, y = 0\end{aligned}$$

- The cross-product between the fitted value $\hat{y}$ and the residual vector r is zero, i.e.

$$\hat{y}^T\ r = \hat{\beta}^T X^T\ r = 0$$

This implies that the residual vector r is **orthogonal to each column of X and $\hat{y}$**.

2.1.4 The Hat Matrix Properties

1. Let c be any linear combination of the columns of $\mathbf{X}$, then

$$\mathbf{H}c = c$$

This is true since

$$\mathbf{HX} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X} = \mathbf{X}.$$

2. *Symmetric*
We can prove this using the definition of $\mathbf{H}$. Indeed,

$$\mathbf{H}^T = (\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T)^T = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T = \mathbf{H}.$$

3. *Idempotent*, i.e. $\mathbf{HH} = \mathbf{HH}^T = \mathbf{H^TH} = \mathbf{H}$.
Indeed,

$$\mathbf{HH} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T = \mathbf{H}$$

This also implies that $\mathbf{H}(\mathbf{I} - \mathbf{H}) = \mathbf{0}_{n\times n}$.

4. $trace(\mathbf{H}) = p$, that is the number of LS coefficients we estimated.
We can easily prove that as follows:

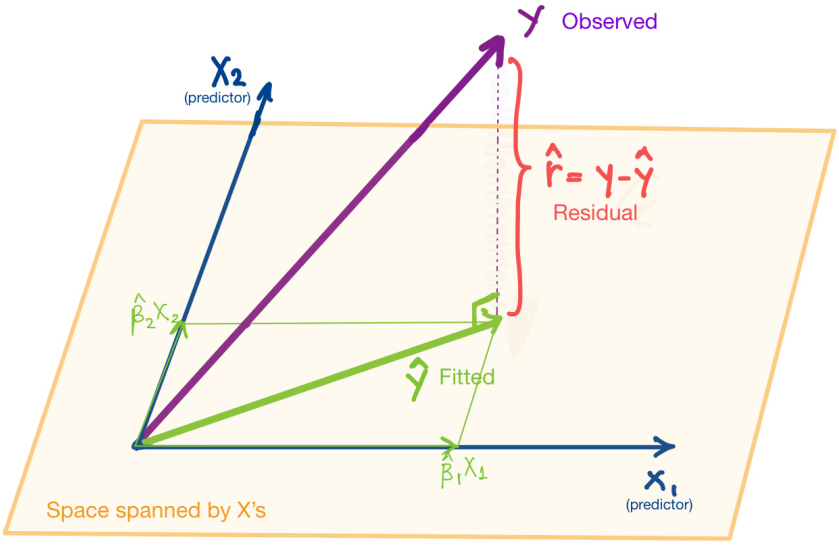
$$trace(\mathbf{H}) = trace(\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T) = trace(\mathbf{X}\mathbf{X}^T(\mathbf{X}^T\mathbf{X})^{-1}) = trace(\mathbf{I}_{p\times p}) = p$$

Here we use the property that $trace(AB) = trace(BA)$.

2.1.5 Geometric Representation of LS

When $p = 1$, the linear regression is represented as a straight line in two dimensions. When $p = 2$, the fitted regression equation corresponds to a plane in three dimensions. When $p > 2$, the fitted regression line is a *hyperplane*, the generalization of a p-dimensional plane in a $(p + 1)$-dimensional space.

Let us focus on the case of 2 predictors x_1, x_2 and consider the representation below:



Observe that the fitted $\hat{\mathbf{y}} = \mathbf{X}\hat{\beta}$ can only range over vectors in the subspace spanned by the columns of $\mathbf{X}$. This is because $\hat{\mathbf{y}}$ is obtained by minimizing the Euclidean distance between the vectors $\mathbf{y}$ and $\hat{\mathbf{y}}$. In fact, $\hat{\mathbf{y}}$ is the **projection** of $\mathbf{y}$ onto the space spanned by the columns of $\mathbf{X}$.

We have the following two spaces:

- Estimation Space
  - The estimation space is the space spanned by the columns of $\mathbf{X}$ and is a p-dimensional subspace in $\mathbb{R}^n$. This is a subspace that consists of vectors that can be written as *linear combinations of the columns of* $\mathbf{X}$.
  - The LS squares estimator $\hat{\beta}$ is an element in the estimation space.
  - $\hat{y}$is the *projection* of y onto the estimation space, since it is obtained by minimizing the Euclidean distance between the vectors $\mathbf{y}$ and $\hat{\mathbf{y}}$, i.e. $||y - \hat{y}||^2$.

- $\mathbf{H}_{n\times n}$ is *projection/hat matrix* of the estimation space and is symmetric, unique, and idempotent.

- Error Space
  - The error space is an $(n-p)$-dimensional space that is orthogonal to the estimation space. The *projection matrix* of the error space is $(\mathbf{I}-\mathbf{H})$.
  - The residual $\mathbf{r}$ is the *projection* of $\mathbf{y}$ onto the error space, and is **orthogonal** to the estimation space. This implies that $\mathbf{r}$ is orthogonal to any vector in the estimation space, including each column of $\mathbf{X}$.
  - When the intercept is included in the model, then

$$\sum_{i=1}^{n} r_i = 0$$

In general, $\sum_{i=1}^{n} r_i X_{ij} = 0$, $j = 1, \ldots, p$ due to the normal equations.

2.1.6 Rank deficiency

We discussed that the design matrix $\mathbf{X}$ is an $n \times p$ matrix. If this matrix is **not of full rank** (i.e., its columns are not linearly independent), the matrix $\mathbf{X^T X}$ *cannot be inverted (singular matrix)*. This means that the solution to the *Normal Equations* is **not** unique. In Statistics, we call this an **identifiability problem**, because we cannot uniquely define estimators for the unknown parameters.

Fortunately, R can cope well with this problem. To solve the LS equations R uses so-called **QR decomposition** method. This method is described below for completeness, but it is beyond the scope of the course.

Birthweight Example

If we want to extract the matrix of a fitted model in R we have:

```r
head(model.matrix(birthweight.mlr1))
```

```
##   (Intercept) Length Headcirc Gestation smoker mage mnocig mheight mppwt fage
## 1           1     56       34        44      0   20      0     162    57   23
## 2           1     53       36        40      0   19      0     171    62   19
## 3           1     58       39        41      0   35      0     172    58   31
## 4           1     53       38        44      0   20      0     174    68   26
## 5           1     54       37        42      0   24      0     175    66   30
## 6           1     51       38        38      0   29      0     165    61   31
##   fedyrs fnocig fheight lowbwt
## 1     10     35     179      0
## 2     12      0     183      0
## 3     16     25     185      0
## 4     14     25     189      0
## 5     12      0     184      0
## 6     16      0     180      0
```

We output only the top part of the matrix, since its dimensions are 42×14.

QR Decomposition: How the LS estimates $\hat{\beta}$ are solved in R

Denote the **QR decomposition** (also called QR factorization) of $\mathbf{X}$ as

$$\mathbf{X}_{n\times p} = \mathbf{Q}_{n\times p}\mathbf{R}_{p\times p}$$

where $\mathbf{Q}$ is an orthogonal matrix (i.e. $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}_{p\times p}$) and $\mathbf{R}$ is an upper triangular matrix, i.e. all the entries in $\mathbf{R}$ below the diagonal are equal to 0. Then,

$$\begin{aligned}
\hat{\beta} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y \\
(\mathbf{X}^T\mathbf{X})^{-1} &= (\mathbf{R}^T\mathbf{R})^{-1} = \mathbf{R}^{-1}(\mathbf{R}^T)^{-1} \\
\hat{\beta} &= \mathbf{R}^{-1}\mathbf{Q}\mathbf{y} \\
\mathbf{R}\hat{\beta} &= \mathbf{Q}\mathbf{y}
\end{aligned}$$

The last equation is solved easily via *backsolving* since $\mathbf{R}$ is an upper triangular matrix.

Gram-Schmidt Algorithm

One method for computing the QR decomposition is the **Gram-Schmidt algorithm**. It works as follows:\ Take

$$\mathbf{A}_{n\times p} = [a_1|a_2|\dots|a_p],$$

where a_j denotes the jth column of $\mathbf{A}$. Then, define a sequence of e_i's and $\mathbf{q}_i's$ recursively:

1. $e_1 = a_1$, $\mathbf{q}_1 = \frac{e_1}{||e_1||}$
2. $e_2 = a_2 - (a_2^T\mathbf{q}_1)\mathbf{q}_1$, $\mathbf{q}_2 = \frac{e_2}{||e_2||}$
3. . . .
4. $e_{k+1} = a_{k+1} = \sum_{j=1}^{k}(a_j^T\mathbf{q}_j)\mathbf{q}_j$

The resulting QR decomposition is

$$\mathbf{A}_{n\times p} = [a_1|a_2|\dots|a_p] = \mathbf{A}_{n\times p} = [\mathbf{q}_1|\mathbf{q}_2|\dots|\mathbf{q}_p]\mathbf{R} = \mathbf{QR}$$

It is easy to check that $\mathbf{R}$ is indeed an upper triangular matrix.