3.3 Checking Model Assumptions

In addition to use the residuals to identify outliers, we are also going to use them to identify the following departures from the normal error model regression:

1. The error terms do not have constant variance.
2. The error terms are not normally distributed.
3. The error terms are not independent.
4. The regression function is not linear.

We typically start the analysis using informal diagnostic plots of residuals to provide information on whether any of these types of departures from assumptions. The following plots of residuals (or studentized residuals) will be used for this purpose:

1. Plot of residuals against fitted values.
2. Plot of residuals against predictor variable.
3. Plot of residuals against time or other sequence.
4. Plots of residuals against each individual predictor variables.
5. Normality probability plot of residuals.

When we look at these plots, in general, we try to find systematic patterns as well as large absolute values of residuals.

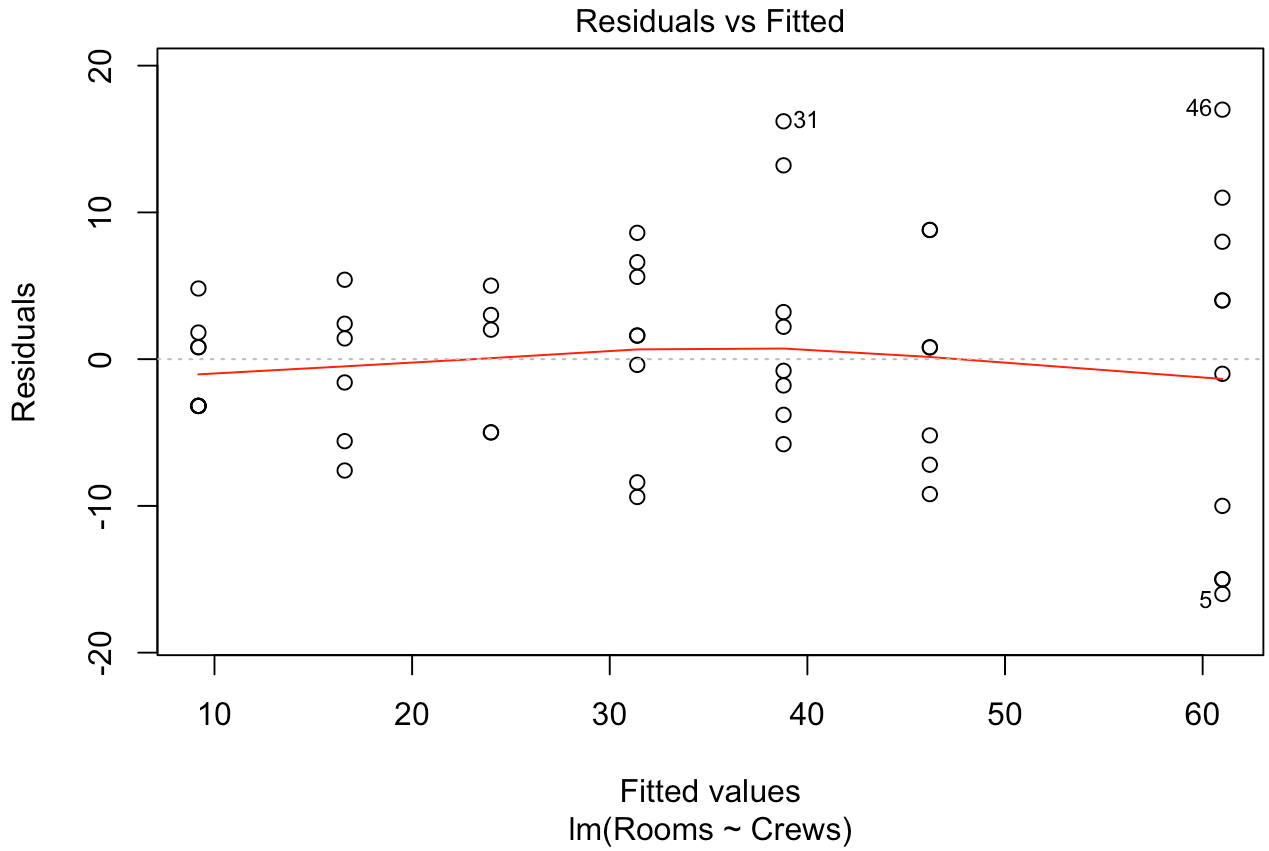
3.3.1 Constant Variance Assumption

We start our analysis with two examples:

Cleaning Example

In the `cleaning.txt` data set we have rooms cleaning by different crews.

```
cleaning = read.table("data/ch3/cleaning.txt", header=TRUE)
cleaning.lm = lm(Rooms ~ Crews, data=cleaning)
plot(cleaning.lm, which=1)
```

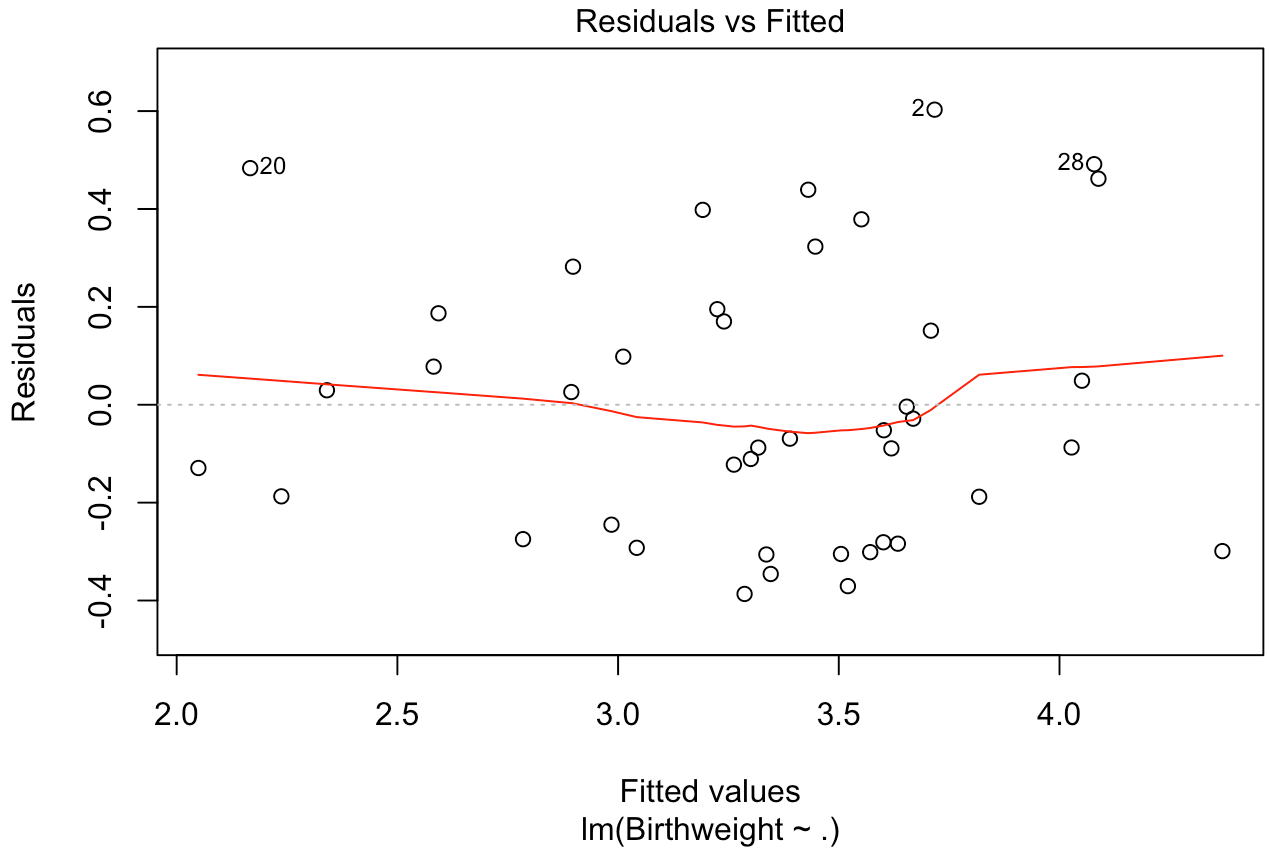


In this plot, we observe that the variance is smaller for lower `crews` numbers and larger for higher `crews` numbers indicating that *the variance is not constant*.

Let us now check whether the constant variance assumption holds in the `Birthweight` example:

Birthweight Example

```
plot(birthweight.mlr1, which=1)
```



Here we can see that the variance seems to be approximately constant, since all the points in the plot seem to nicely fit within two parallel lines.

In general, if the variance is constant, the residuals will look like a football-shaped cloud. So, when we check the residual vs. fitted values plot, we look for a fan type shape or other trends that might reveal departure from the constant variance assumption.

Apart from this diagnostic plot, we also have statistical tests to check for the constant variance assumption. Here we focus on the **Breusch-Pagan test**.

Breusch-Pagan Test

The hypothesis to test is:

$$\begin{cases} H_0: \text{the variance is constant} \\ H_\alpha: \text{the variance is not constant} \end{cases}$$

and the test statistic is formulated as

$$\text{BP} = nR^2$$

where R^2 is the coefficient of Determination between the squared residuals r_i^2 (of a LS regression between Y and variables $X_1, X_2, \ldots, X_p$ – including the intercept) and the covariates (or a sub-set) $X_1, X_2, \ldots, X_p$.

Under the H_0 hypothesis, then

$$\text{BP} \sim \chi^2_{p-1}$$

asymptotically.

The goal of the Breusch-Pagan test is to check whether the variance of the errors from a regression is **dependent** on the *independent variables*. Under the H_0 hypothesis, the variance is constant, i.e. there is *homoscedasticity*. Under H_α, we have *heteroscedasticity*, i.e. not constant variance.

In `R`, to perform the Breusch-Pagan test we use `bptest` in package `lmtest`.

Birthweight Example

```
library(lmtest)
bptest(birthweight.mlr1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  birthweight.mlr1
## BP = 5.6533, df = 13, p-value = 0.9579
```

Since the p-value is greater than $\alpha = 5\%$, we fail to reject the H_0 and conclude that *the variance is constant*.

Variance Stabilizing Transformations

The *goal* of a variance stabilizing transformation is to find a function (transformation) of the response, $h(Y)$, to achieve constant variance.

Here is how it works!

Suppose h is a smooth function. Using Taylor's theorem, the expansion of $h(Y)$ around $\mathbf{E}(Y)$ is:

$$h(Y) = h(\mathbf{E}(Y)) + h'(\mathbf{E}(Y))(Y - \mathbf{E}(Y)) + \text{ Remainder}$$

The remainder is assumed *small with high probability* and we can ignore it. So,

$$Var(h(Y)) \approx \left(h'(\mathbf{E}(Y))\right)^2 Var(Y)$$

We haven't determined $h(\cdot)$ yet. In fact, we want to choose a transformation h such that $Var(h(Y))$ is approximately constant.

Examples of Transformations

1. Suppose that the variance of Y is proportional to the mean of Y, i.e.,

$$Var(Y) \propto \mathbf{E}(Y)$$

If we select h such that:

$$h'(z) = \frac{1}{\sqrt{z}} \Rightarrow h(z) \propto \sqrt{z}$$

then plugging-in the value of $h'(z)$ evaluated at $\mathbf{E}(Y)$ in the variance of $h(Y)$ equation, the variance of $h(Y)$ will be approximately constant. Indeed,

$$Var\left(\sqrt{Y}\right) \approx \left(\frac{1}{\sqrt{\mathbf{E}(Y)}}\right)^2 Var(Y) = \frac{Var(Y)}{\mathbf{E}(Y)} \approx const.$$

2. Suppose that the variance of Y is proportional to the squared mean of Y, i.e.,

$$Var(Y) \propto (\mathbf{E}(Y))^2.$$

If we select h such that:

$$h'(z) = \frac{1}{z} \Rightarrow h(z) = \log(z)$$

this leads to

$$Var(\log Y) \approx \frac{1}{(\mathbf{E}(Y))^2} Var(Y) \approx const.$$

Question: So, the question here is how do we find such transformations in *practice*?

Answer: We get an *idea* of the underlying relationship if we find the relation between the Residual Variance, i.e. $Var(Y) = Var(\varepsilon)$ and the Fitted Values, i.e the estimated $\mathrm{E}(Y)$ and we can do so by employing *residual plots*.

A summary of variance stabilizing transformations:

1. When $Var(\varepsilon) \propto \mathbf{E}(Y)\}$, then $h(Y) = \sqrt{Y}$. This is a transformation that is typically suitable for *counts* from the Poisson distribution.

2. When $Var(\varepsilon) \propto (\mathbf{E}(Y))^2$, then $h(Y) = \log(Y)$ or $\log(Y+1)$. This is a transformation that is suitable for data whose range of Y is very broad, e.g., from 1 to several thousands; suitable for estimating percentage effect ($Y \propto CX^{\alpha}$.)

3. When $Var(\varepsilon) \propto (\mathbf{E}(Y))^4$, then $h(Y) = 1/Y$ or $1/(Y+1)$. This is a transformation that is suitable for data where Y measures the waiting time or survival time. Taking reciprocals changes the scale from time (time per response) to rate (response per unit time).

3.3.2 Normality Assumption

If we have a sample $z_1, z_2, \ldots, z_n$ and we wish to examine the hypothesis that the z's are a sample from a normal distribution with mean μ and variance σ^2, then the easiest way to check for normality is by performing two graphical tests:

a. plot the histogram
b. construct a QQ-plot

In the case of MLR, we construct these plots for the residuals (standardized/studentized or plain) of the fitted model.

QQ-plot

Assume that we have a sample $z_1, z_2, \ldots, z_n$ from an unknown distribution F and we want to check whether the z_is are Normally distributed. Then, to construct the QQ-plot, the process is the following:

1. Order the z's: $z_{(1)}, z_{(2)}, \ldots, z_{(n)}$.

2. Compute $u_i = \Phi^{-1}(\frac{i}{n+1})$, where Φ is the cdf of the $N(0,1)$ and i is the order of the data ($i = 1, 2, \ldots, n$).

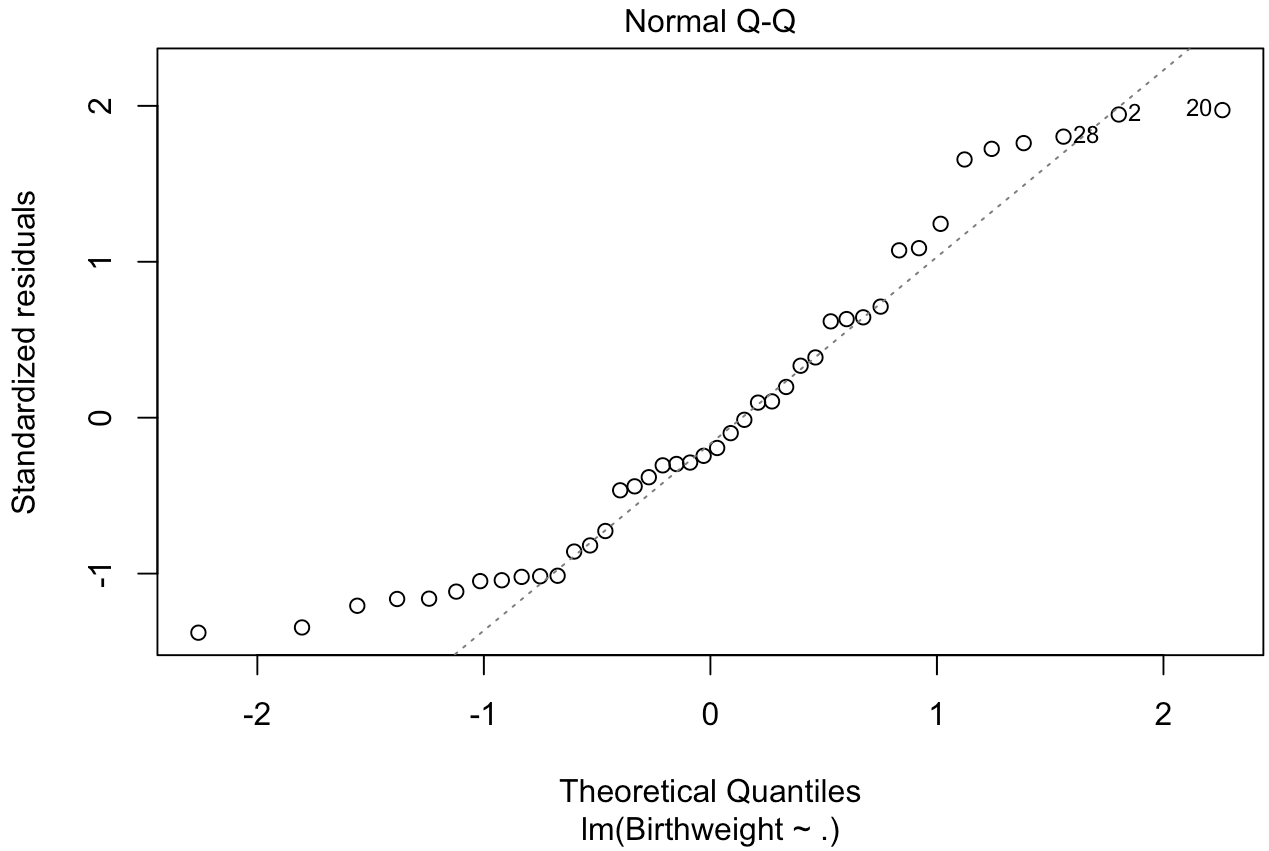
3. Plot $z_{(i)}\}$ against u_i.

Plot Interpretation: If the z's are normally distributed, the points should *approximately* fall on a *straight line*.

The Birthweight Study

We start by looking at the QQ-plot. This is done in `R` as follows:

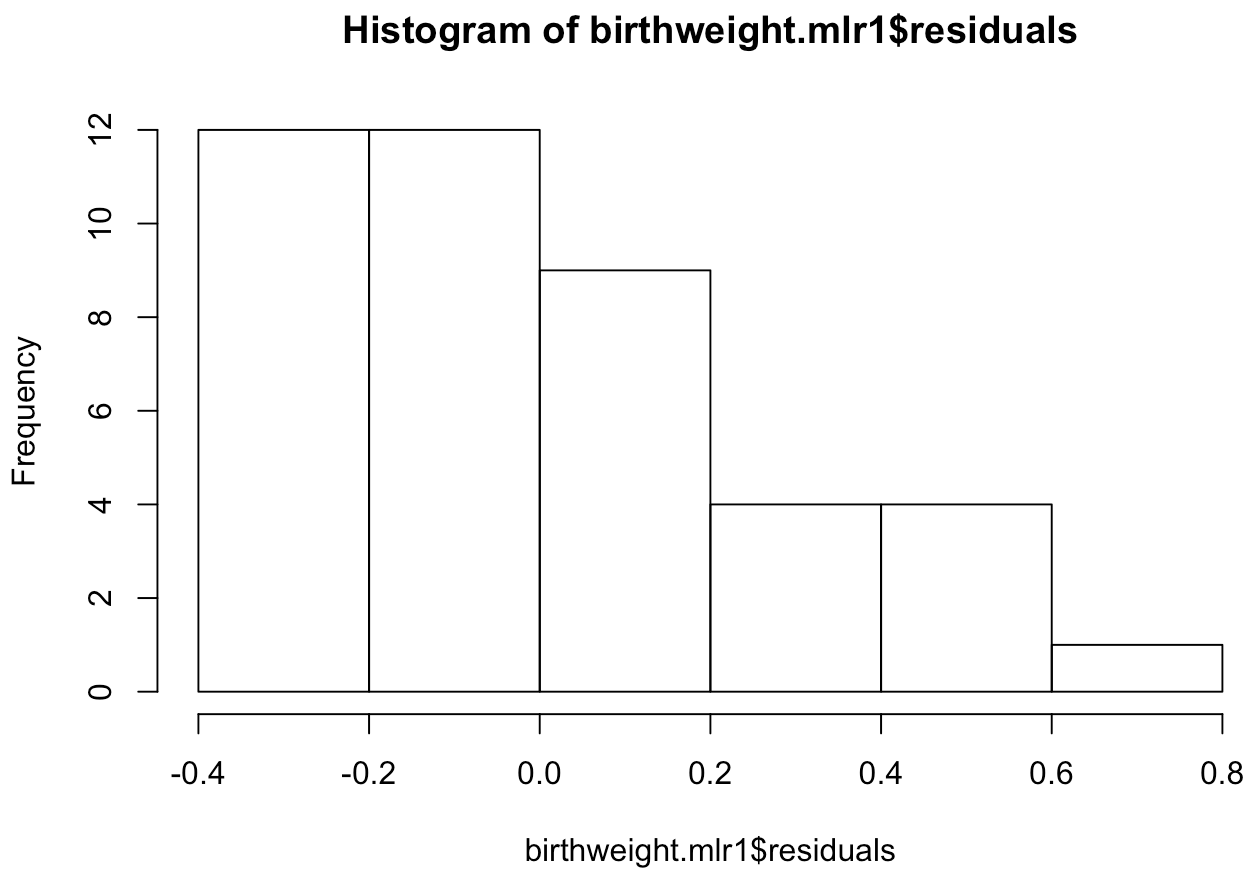
```
plot(birthweight.mlr1, which=2)
```



It seems that the points in the QQ-plot do not fall on a straight line. In fact, we obsevre that the points in the left hand side and the right hand side are off the line indicating a skewnes in the distribution of the residuals.

We can also plot the histogram of the residuals using the `hist()` function

```
hist(birthweight.mlr1$residuals)
```



In this plot, it is also clear that the distribution of the data is right skewed.

Apart from the residual plots, we also have statistical tests that allow us to test for the Normality assumption. Here we discuss two of these: (i) the Shapiro-Wilk test and (ii) the Kolmogorov-Smirnov test.

Shapiro-Wilk Normality Test

This test is good for $n \leq 50$. The hypothesis to test is:

$$\begin{cases} H_0: \text{the distribution is normal} \\ H_\alpha: \text{the distribution is not normal} \end{cases}$$

and the test statistic is formulated as

$$W = \frac{\left(\sum_{i=1}^{n} a_i r_{(i)}\right)^2}{\sum_{i=1}^{n}(r_i - \bar{r})^2},$$

where $r_{(i)}$ is the ith largest value of the r_i's and the a_i terms are calculated using the means, variances, and covariances of the r_is. Small values of W will lead to rejection of the null hypothesis.

Kolmogorov-Smirnov Normality Test

This test is good for $n > 50$. The hypothesis to test is:

$$\begin{cases} H_0: \text{the distribution is normal} \\ H_a: \text{the distribution is not normal} \end{cases}$$

and the test statistic is formulated as

$$D_n = \sup_x \left|F_n(x) - \Phi(x)\right|$$

where $\Phi(x)$ is the cdf of the Normal and F_n the empirical distribution function F_n for n i.i.d. ordered observations X_i is defined as

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{[-\infty, x]}(X_i)$$

Intuitively, the KS statistic takes the largest absolute difference between the two distribution functions across all x values.

The Shapiro-Wilk test is done by using the `shapiro.test` function in `R`, while the Kolmogorov-Smirnov test is performed by calling the `ks.test` function in `R`.

Birthweight Example

```
shapiro.test(birthweight.mlr1$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  birthweight.mlr1$residuals
## W = 0.93365, p-value = 0.01724
```

The p-value of 0.01283 is less than the significance level $\alpha = 0.05$. So, we reject the null hypotheses of normality and conclude that the normality assumption is *not* satisfied.

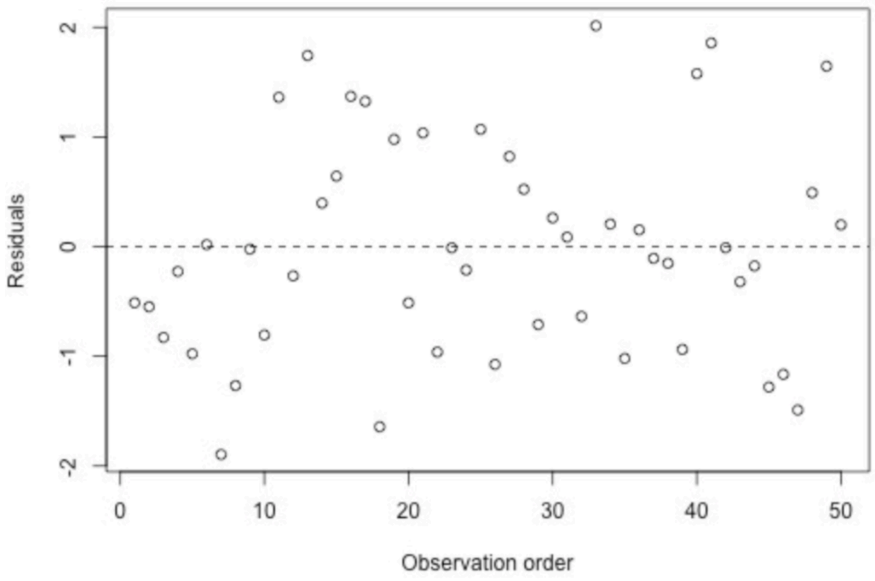
3.3.3 Independence of Errors Terms Assumption

The next assumption we want to check is the **independence** of the residual terms. When the independence assumption is violated, this implies that we have correlation is typically *temporal* or *spatial*.

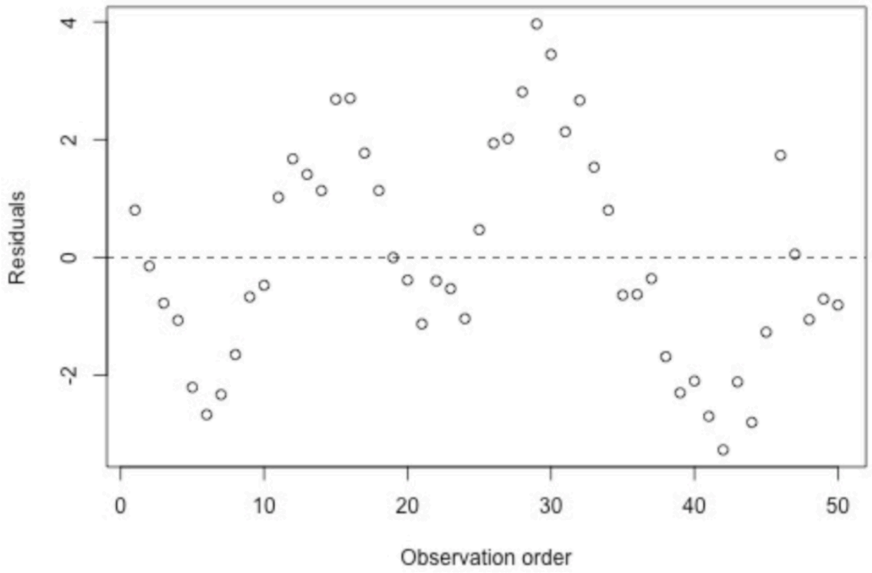
To check these assumptions, we plot the *residuals* against *time* or another *index* and we create the so-called sequence plot. To discover whether the independence assumption is violated we check whether data above or below the mean tend to be followed by data above or below the mean.

Sequence Plot Examples

Below you can see the sequence plots from two different data sets:



This plot shows that the residuals are randomly scattered about the mean, which means that there are no concerns for serial correlation.



This plot, on the other hand, shows that the residuals follow a sinusoidal pattern, which implies that we have serial dependence in the data that we need to account for.

We also have a formal test statistic, that helps us to detect correlation

Durbin-Watson Test

The Durbin-Watson test statistic is

$$DW = \frac{\sum_{k=1}^{n-1}(r_k - r_{k+1})^2}{\sum_{k=1}^{n} r_k^2}$$

The decision rule is the following: *"If $DW < 2$, then there is evidence for positive serial dependence."*

In `R`, we perform a correlation test by using the `dwtest` function in the `lmtest` library.

3.3.4 Linearity Assumption

The linearity assumption is a model structure assumption. But, how do we check whether

$$\mathrm{E}(y) = \mathbf{X}\beta$$

is correct?

In the *Simple Linear Regression*, this is an easy task to do: we look at the scatter plot and we check whether the data are scattered around the fitted regression line.

So, one idea here would be that we create all pairwise scatter plots between the response and one of the predictors. However, if a MLR model is appropriate, these plots would be quite noise. Specifically, there will be a lot of variation about the regression line between Y and X_i that will *not* allow us to draw conclusions about the linearity. In addition to that, it does not allow us to incorporate interactions between predictors.

For that purpose, we use Partial Regression plots that are also called Added Variable plots[13].

Partial Regression Plots

We want to know the relationship between the response Y and a predictor X_i *after the effect of the other predictors has been removed*. To remove the effect of the other predictors, we run the following **two** regression models:

$$Y \sim X_1 + \ldots + X_{i-1} + X_{i+1} + \ldots \qquad \text{(Model 1)}$$
$$X_i \sim X_1 + \ldots + X_{i-1} + X_{i+1} + \ldots \qquad \text{(Model 2)}$$

Then, we obtain the residuals from each model:

$$\mathbf{r}_y = \text{residuals from Model 1}$$
$$\mathbf{r}_i^X = \text{residuals from Model 2}$$

and plot them against each other, i.e. $\mathbf{r}_y$ vs. $\mathbf{r}_k^X$. For a valid model, the added-variable plot should produce **points randomly scattered around a line through the origin with slope $\hat{\boldsymbol{\beta}}_k$**.

Transformations to overcome Non-Linearity

If the lack-of-linearity is the departure from the model assumptions, then we perform **linearizing transformations**. Such examples are presented below:

1. Use $\log(Y)$ vs. $\log(X)$, i.e. apply logarithm to the response and the predictors.
   This is suitable when $\mathrm{E}(Y) = \alpha X_1^{\beta_1} \ldots X_p^{\beta_p}$.

2. $\log(Y)$ vs. X, i.e. apply logarithm to the response only.
   This is suitable when $\mathrm{E}(Y) = \alpha \exp \sum_j X_j \beta_j$.

3. $1/Y$ vs. X, i.e. take the inverse of the response. This is suitable when $\mathrm{E}(Y) = \frac{1}{\alpha + \sum_j X_j \beta_j}$.

Birthweight Example

We have already fitted the **full model** that contains all the predictors, we called that `birthweight.mlr1`. Now, we want to create added variables/partial regression plots, one for each of the predictors in the model. Here, we illustrate this method for two of the predictors: `Headcirc` and `Gestation`.

(a) Checking *linearity* with respect to `Gestation` :

First, we remove the predictor `Gestation` from the model, and we store the residuals:

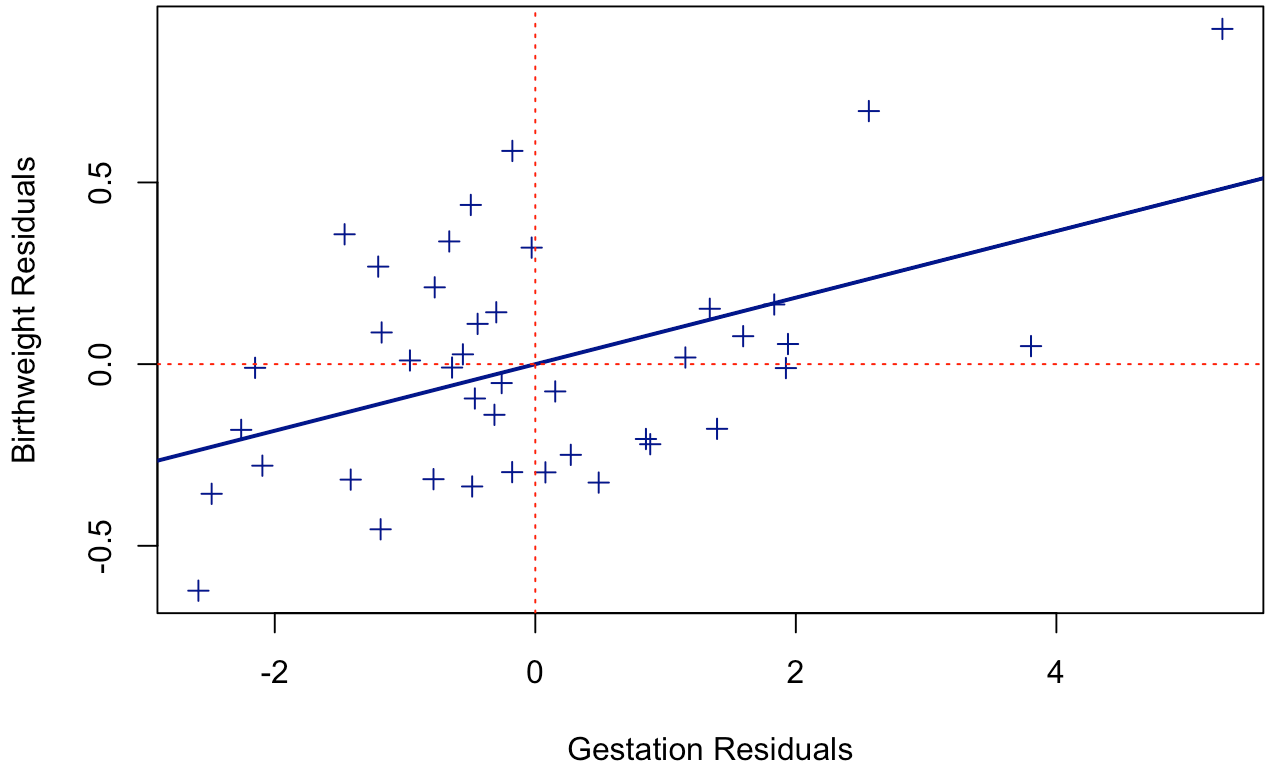
```
y.Gestation = update(birthweight.mlr1, .~. -Gestation)$res
```

Remark: In `R` , we can do this in two ways. Either by re-fitting the model without the `Gestation` variable, or by using the update function in `R` that does exactly that for us.
Then, we fit a MLR model with `Gestation` as the response against the remaining predictors, and we store the residuals:

```
x.Gestation = lm(Gestation ~ ., data = birthweight2[,-c(2)])$res
```

Remark: In `R` : when we fit the model, we need to make sure that we **remove** the response from the predictors.
Finally, we create the _partial regression plot:

```
plot(x.Gestation, y.Gestation, xlab="Gestation Residuals", ylab="Birthweight Resi
abline(lm(y.Gestation ~ x.Gestation), col='Darkblue', lwd=2)
abline(v = 0, col="red", lty=3)
abline(h = 0, col="red", lty=3)
```

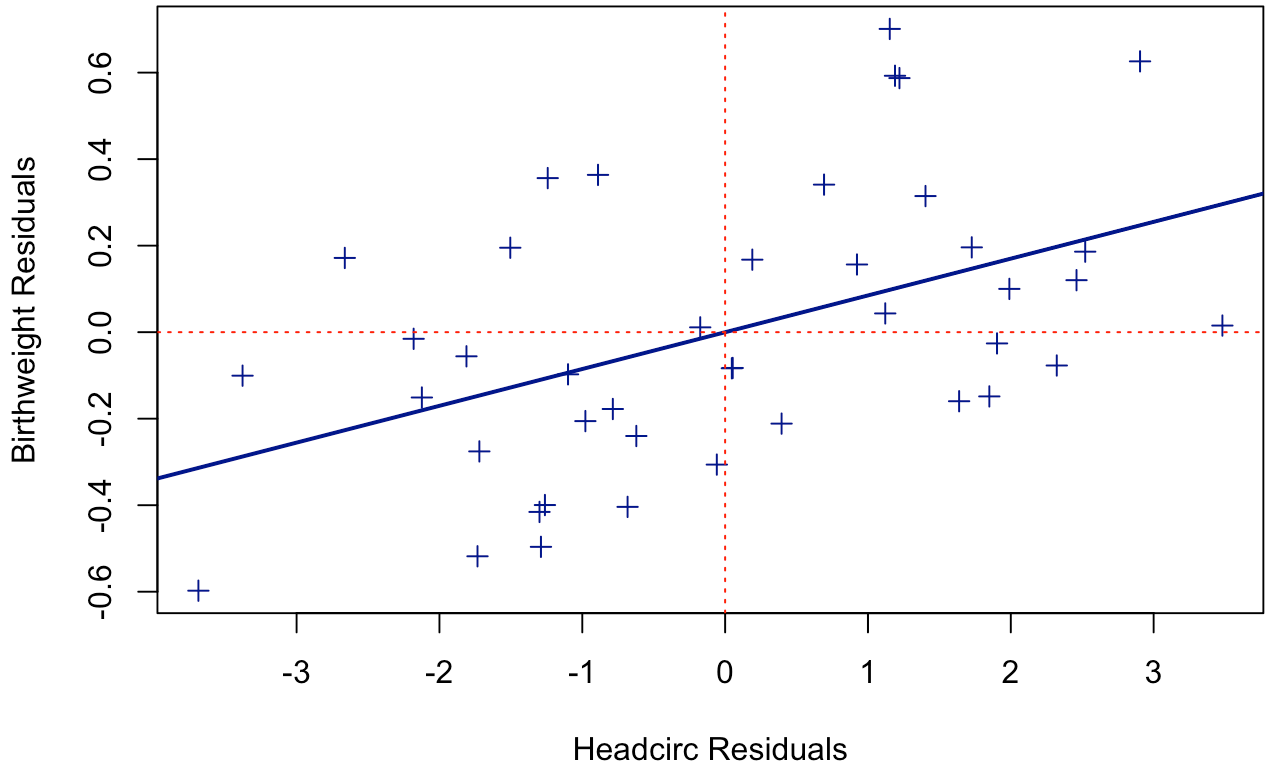


The points appear to be randomly scattered around the regression line between r_y and $r^X_{Gestation}$. This means that there is a linear relationship between `Birthweight` and `Gestation`.

(b) Checking *linearity* with respect to `Headcirc`.

We repeat the same process to check linearity between `Birthweight` and `Headcirc`:

```
y.Headcirc = update(birthweight.mlr1, .~. -Headcirc)$res
x.Headcirc = lm(Headcirc ~ ., data = birthweight2[,-c(2)])$res
plot(x.Headcirc, y.Headcirc, xlab="Headcirc Residuals", ylab="Birthweight Residua
abline(lm(y.Headcirc ~ x.Headcirc), col='Darkblue', lwd=2)
abline(v = 0, col="red", lty=3)
abline(h = 0, col="red", lty=3)
```



Once again, the points appear to be randomly scattered around the fitted regression line, so the linearity assumption is also satisfied here. We should continue the same process with *all the remaining variables in the model*. If *all* plots yield the same results, then we can conclude that the linearity assumption is satisfied for the MLR model.

3.3.5 Box-Cox Transformations

The Box-Cox method was initially introduced to remedy departures from the normality assumption in linear models. The idea behind is to transform the response (Y) with a **power transformation** that will result in a *normally distributed transformed* Y. It turns out that in doing this, it often reduces non-linearity as well.

Box-Cox Transformations

The proposed family of transformations is[14]:

$$g_\lambda(Y) := \begin{cases} \frac{Y^\lambda - 1}{\lambda}, & \lambda \neq 0 \\ \log(Y), & \lambda = 0 \end{cases}$$

The goal is to find λ (if possible) for which Y^λ follows a Normal distribution.Then, given λ, we estimate the remaining parameters as usual.

How to select λ?

We want to choose a λ that *maximizes* the likelihood of the data, under the assumption that the *transformed data* $g_\lambda(\mathbf{y})$ will have a normal distribution. This means that

$$g_\lambda(\mathbf{y}) = \mathbf{X}\beta + \varepsilon, \quad \varepsilon \sim N_n(\mathbf{0}, \sigma^2\mathbf{I})$$

If we write the *log*-likelihood function for $\lambda \neq 0$, we obtain:

$$\ell(\lambda) = -\frac{n}{2}\log(RSS_\lambda/n) + (\lambda - 1)\sum_{i=1}^{n}\log(y_i)$$

where RSS_λ is the Residual Sum of Squares for the **transformed** Y, i.e. when $g_\lambda(\mathbf{y})$ is the response. When $\lambda = 0$ the *log*-likelihood becomes:

$$\ell(0) = -\frac{n}{2}\log(RSS_0/n) - \sum_{i=1}^{n}\log(y_i)$$

The second term in the log-likelihood function corresponds to the *Jacobian of the transformation*[15].

Remark: In this framework, it *does not make sense* to choose λ just by minimizing RSS_λ. The reason is that for each λ, the Residual Sum of Squares are measured in a different scale, which means that it is not possible to directly compare them. Therefore, the method accounts for that by looking at the log-likelihood instead.

In R , we can plot the log-likelihood as a function of λ ($L(\lambda)$) versus λ. Theoretically, λ can be aby real number. However, in practice, it has been observed that this method works well for $\lambda \in (-2, 2)$. Also, in practice, it is also common to round $\hat{\lambda}$ to a nearby value like:

$$-1, \ -0.5, 0, 0.5, \ \text{or } 1$$

In this way, the transformed Y is much easier to interpret.

Using this method, we can also construct a hypothesis test on whether we really need the transformation g_λ:

$$\begin{cases} H_0 : \lambda = 1 \\ H_\alpha : \lambda \neq 1 \end{cases}$$

Under the H_0, we know (Box-Cox proved) that

$$2(L(\hat{\lambda}) - L(\lambda_0)) \sim \chi^2_1$$

Instead of the test, we can also construct a Confidence Interval for λ as follows:

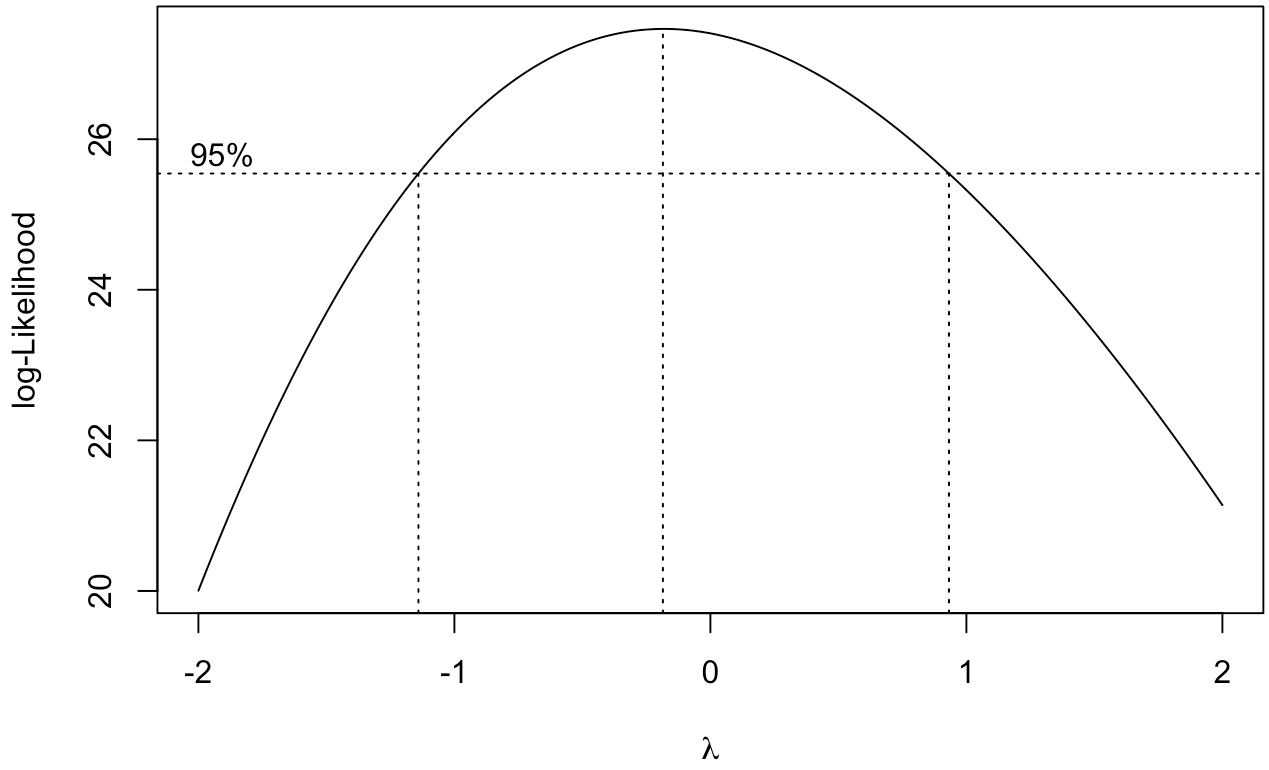
$$\left\{ \lambda : L(\lambda) > L(\hat{\lambda}) - \frac{1}{2}\chi^2_1(1-\alpha) \right\}$$

To perform a Box-Cox transformation in R , we use the `boxcox` function in the `MASS` library. In the corresponding plot, the 95% confidence interval for λ is also shown.

Birthweight Example

In the `Birthweight` example, we found that the normality assumption is **not** satisfied. Therefore, we check whether a transformation of the response can fix the departure from normality:

```
library(MASS)
birthweight.transformation = boxcox(birthweight.mlr1, lambda=seq(-2,2, length=400
```



If we want to extract the exact λ computed by the method, we have

```
lambda <- birthweight.transformation$x[which.max(birthweight.transformation$y)]
lambda
```

```
## [1] -0.1854637
```

Indeed, based on this output, we can say that a value of λ near -0.1854637 is would probably fix the departure from the normality assumption. As we discussed above, we often want to select a λ that corresponds to a tranformation that we can eaily interpret. If we look at the 95% interval for λ, we observe that both $\lambda = 0$ and $\lambda = 1$ are included in the interval. So, both these values could be used as transformations of Y. We prefer to choose the one that is closer to the optimal λ unless for some reason it is not applicable in our case. Here, $\lambda = 0$ is selected, which means that we will perform a `log` transformation on Y.

3.3.6 Permutation Tests

Many times in practice, we have no information about the underlying distribution of the data, or in the case of MLR we the normality assumption fails, in which case we cannot use the usual hypothesis tests and p-values to draw meaningful conclusions. Therefore, in cases where we are uncertain of the distribution of the test statistics, we use **permutation tests**.

Permutation tests are not performed similar to the t or F tests that we have seen, but are based on estimating the *empirical distribution of the data*. Thus, they are computational in nature and their calculation is based on the fact that we can replicate/simulate data from the "true" model. To this end, we use a powerful technique that is called Monte Carlo method.

Permutation Test Overview

By definition, a test statistic is a function of the data; denoted by $g(data)$. In a hypothesis test, the test statistic tends to take *extreme values under the alternative hypothesis* H_α.

So, to conduct a permutation test, we need to

1. *evaluate* the test statistic on the **observed** data, denote this by g_0.
2. *estimate* the distribution of $g(data)$, when the data are generated under the H_0.
3. *calculate* the p-value defined as

$$\mathrm{P}\Big(g(data) \text{ is more extreme than the observed } g_0 \,|\, \text{ data } \sim H_0\Big)$$

Observe that the p-value (as always) is a conditional probability *given the* H_0 hypothesis. So, if we know how to generate data under the null, then we can compute this probability numerically via the Monte Carlo method.

In a nutshell, the Monte Carlo method allows us to generate many samples under the H_0, and then if we compute our test statistic of interest for each of these samples and compare it with the test statistic computed for the true data set, then Monte Carlo says that the p value will be

approximated by

$$\frac{\text{no. of times } g(\text{sample data}) > g(\text{data under } H_0)}{\text{total no. of generated samples}}$$

for a *large* number of samples.

Let's briefly discuss why this is true, and what is the main ingredient in Monte Carlo:

Monte Carlo Method Basics

Suppose the pdf (or pmf) of a random variable Y does not have a simple form, therefore it is not easy to calculate $\mathrm{E}(Y)$ *explicitly*. But suppose it is easy to write a short R script to generate such a random variable, i.e. it is easy to simulate it. In this case, we can obtain an *approximation* of $\mathrm{E}(Y)$ as follows:

1. Generate $N = 1000$ samples from this distribution, $Y_1, \ldots, Y_N$,
2. Approximate the mean by

$$\mathrm{E}(Y) \approx \frac{1}{N}\sum_{i=1}^{N} Y_i$$

That is, the **population** mean $\approx$ **sample** mean, when N is large (remember Law of Large Numbers?).

This method is so powerful that it also works if we want to approximate the expected value of a *function* of a random variable, that is

$$\mathrm{E}(f(Y)) \approx \frac{1}{N}\sum_{i=1}^{N} f(Y_i)$$

As an example, we can use Monte Carlo to compute the variance of a r.v.

$$Var(Y) = \mathrm{E}\left(Y^2\right) - (\mathrm{E}(Y))^2$$

or a **probability**, since

$$\mathrm{P}(Y > a) = \mathrm{E}\left(\mathbf{1}_{\{Y>a\}}\right),$$

where $\mathbf{1}_{\{\cdot\}}$ is the indicator function.

Permutation Test in R

To see how this works in practice, let's go back to our `Birthweight` example.

Birthweight Example

Assume that we fit the following model:

```
birthweight.mlr.p = lm(Birthweight~Headcirc + Gestation + smoker + Length + mheig
```

and that we want to test the following hypothesis

$$\begin{cases} H_0 : Birthweight \sim Headcirc + Gestation \\ H_\alpha : Birthweight \sim Headcirc + Gestation + Length + mHeight \end{cases}$$

or in other words

$$\begin{cases} H_0 : \beta_{Length} = \beta_{mHeight} \\ H_\alpha : \text{at least one of } \beta_{Length}, \beta_{mHeight} \text{ not equal to } 0 \end{cases}$$

Under H_0, `Length` and `mheight` are **not useful** in explaining the variation in the response `Birthweight`.

Recall that under H_0, `Length` and `mHeight` are **irrelevant**. In our data, we have 42 pairs of `(Length, mHeight)`, each corresponding to a `Birthweight`. What if we **randomly** assign a pair of `(Length, mHeight)` associated with one child to another child's `Birthweight` (repeat this for all 42 pairs), and then re-fit the full model? The inference based on this new data set should not be affected that much, if `(Length, mHeight)` is **truly irrelevant** to `Birthweight`. This is the main idea behind **permutation tests**.

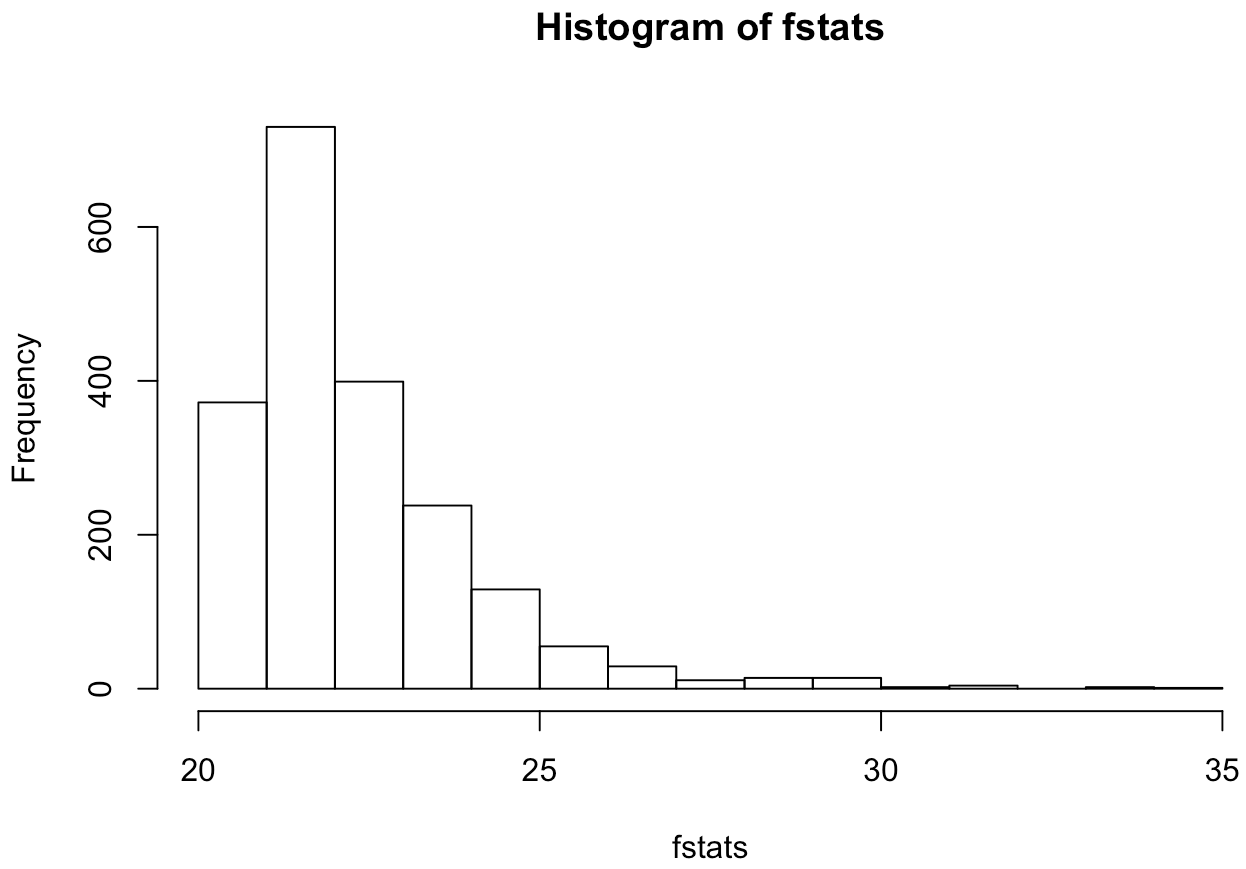
Every pair of values `Length`, `mheight` for a particular newborn can be assigned to any other newborn, assuming that these variables do not have an effect on the response. So, we can fit a full MLR model for each permutation and obtain *many* values of the F-statistic under H_0 (loop in code). Then, we can estimate the p-value associated to the observed data using the Monte Carlo method.

```
n.iter = 2000; # number of times we are going to repeat this re-shuffling

fstats = numeric(n.iter); # initialization for the vector that will store the f-s

# Main for-loop

for(i in 1:n.iter){
  newbirthweight = birthweight2; # define the data set to use

  # permute the values in the columns of the data frame that we want to test out
  # here these variables are Length and  mheight and they are found in columns 1
  # the function sample() takes care of the permutation
  newbirthweight[, c(1,8)] = birthweight2[sample(42), c(1,8)];

  # Compute the test statistic for this new data set
  # i.e. fit the model and extract the fstat
  model = lm(Birthweight ~ Headcirc + Gestation + Length + mheight, data=newbirth
  fstats[i] = summary(model)$fstat[1] # this is a vector that contains the 2000 f
}

# Compute the p-value
# = ( number of fstats > [fstat of original data] )/ total number of fstats
length(fstats[fstats > summary(birthweight.mlr.p)$fstat[1]])/n.iter


## [1] 1
```

Here our p-value is approximately equal to 1 which means that we **fail** to reject the null in favor of the alternative. Therefore, we choose to work with the reduced model instead of the full.

The code above not only outputs the actual p-value, but also the entire *empirical distribution of the test statistic*. This can be seen by plotting a histogram of the `fstat` vector:

```
hist(fstats)
```



3.3.7 Collinearity

In this last section of diagnostics, we are going to investigate how we can detect *collinearity* in our data.

Consider a MLR model with a design matrix $\mathbf{X}_{n\times p}$, including the intercept. If *each* column of $\mathbf{X}$ is **orthogonal** to the other (i.e., the sample correlation of any two predictors is equal to 0), then the least-squares estimators can be greatly simplified:

$$\hat{\beta}_j = [(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}]_\mathbf{j} = \frac{\mathbf{X}_{.j}^\top\mathbf{y}}{||\mathbf{X}_{.j}||^2}$$

where $\mathbf{X}_{.j}$ denotes the j-th column of $\mathbf{X}$. In other words, in this case (only) the least-squares regression coefficient for the j-th predictor *does not depend on whether other predictors are included in the model or not*.

Of course, this is an extreme situation that rarely appears in practice. In applications, we often encounter problems in which many of the predictors are *highly correlated*. In such cases, the values and sampling variance of regression coefficients can be highly dependent on the particular predictors chosen for the model.

Exact Collinearity

If there exists a set of constants $c_1, c_2, \ldots, c_p$ (at least one of them is non-zero), such that the corresponding linear combination of the columns of $\mathbf{X}$ is zero, i.e.:

$$\sum_{j=1}^{p} c_j\mathbf{X}_{.j} = \mathbf{0},$$

then the columns of $\mathbf{X}$ are called *linearly dependent* and there is **exact collinearity**. That is, at least one column in the design matrix X can be expressed as a linear combination of other columns. In case,also, the column space of $\mathbf{X}$ has dimension $< p$.

The implications of *exact collinearity* are that:

1. $(\mathbf{X}^\top\mathbf{X})^{-1}$ does not exists.
2. the LS estimator $\hat{\beta}$ is not unique, and
3. the corresponding linear model is not identifiable.

Example

Suppose the 1st column of $\mathbf{X}$ is the intercept, and the 2nd column of $\mathbf{X}$ is the vector $(2, 2, \ldots, 2)^\top$.

Then, if

$$(\hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3, \ldots)^\top$$

is one LS estimator of β, the vector

$$(\hat{\beta}_1 - c, \hat{\beta}_2 + c/2, \hat{\beta}_3, \ldots)^\top$$

is also an estimator of β, where c is any real number.

We generally do not need to worry about exact collinearity, since R can detect it and fix it automatically.

But, we **do** worry about...

Approximate Collinearity

We have approximate collinearity when at least one column $\mathbf{X}_{,j}$ can be approximated by the others:

$$\mathbf{X}_{.k} \approx -\sum_{j \neq k} c_j \mathbf{X}_{.j} / c_k$$

A simple *diagnostic* for this is to obtain the regression of $\mathbf{X}_{.k}$ on the remaining predictors, and if the corresponding R_k^2 is close to 1, then we would diagnose approximate collinearity.

Why approximate collinearity is a problem?

In a multiple regression $Y = \beta_1 X_1 + \ldots + \beta_p X_p + \varepsilon$, the LS estimate $\hat{\beta}_k$ is *unbiased* with variance:

$$Var(\hat{\beta}_k) = \sigma^2 \left(\frac{1}{1 - R_k^2} \right) \left(\frac{1}{\sum_{i=1}^{n} (x_{ik} - \bar{x}_{.k})^2} \right)$$

where R_k^2 is the R^2 from the regression of $\mathbf{X}_{.k}$ on the remaining predictors. When R_k^2 is close to 1, the variance of $\hat{\beta}_k$ is large. Consequently we have a:

- *large* Mean Square Error
- *large* (inflated) p-value to the corresponding t-test, i.e, we could *miss* a significant predictor.

The quantity $\left(\frac{1}{1 - R_k^2} \right)$ is called the **k-th variance inflation factor (VIF)** and can be used as a metric to detect the presence of collinearity.

Car position Data Example

Car drivers like to adjust the seat position for their own comfort. Car designers would find it helpful to know how different drivers will position the seat depending on their *size* and *age.* Researchers at the HuMoSim laboratory at the University of Michigan collected the following data on **38** drivers:

- `Age` : Drivers age in years
- `Weight` : Drivers weight in lbs
- `HtShoes` : height with shoes in cm
- `Ht` : height without shoes in cm
- `Seated` : seated height in cm
- `Arm` : lower arm length in cm
- `Thigh` : thigh length in cm
- `Leg` : lower leg length in cm
- `hipcenter` : horizontal distance of the midpoint of the hips from a fixed location in the car in mm

To read the data we need to load the `faraway` library and then open the `seatpos` data set.

Let's start by fitting the **full** model with `hipcenter` as a response and everything else as a predictor:

```
# Fit the FULL model
position.full=lm(hipcenter ~ ., seatpos)
summary(position.full)
```

```
##
## Call:
## lm(formula = hipcenter ~ ., data = seatpos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -73.827 -22.833  -3.678  25.017  62.337
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 436.43213  166.57162   2.620   0.0138 *
## Age           0.77572    0.57033   1.360   0.1843
## Weight        0.02631    0.33097   0.080   0.9372
## HtShoes      -2.69241    9.75304  -0.276   0.7845
## Ht            0.60134   10.12987   0.059   0.9531
## Seated        0.53375    3.76189   0.142   0.8882
## Arm          -1.32807    3.90020  -0.341   0.7359
## Thigh        -1.14312    2.66002  -0.430   0.6706
## Leg          -6.43905    4.71386  -1.366   0.1824
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 37.72 on 29 degrees of freedom
## Multiple R-squared:  0.6866, Adjusted R-squared:  0.6001
## F-statistic:  7.94 on 8 and 29 DF,  p-value: 1.306e-05
```

Note that the p-value of the overall F test is very small (1.306e-05), which means that *at least one of the predictors is statistically significant*, but at the same time none of the predictors is statistically significant according to the individual t tests. This is strange and definitely a *red flag* that we may have collinearity issues.

To investigate the presence of high correlated predictors as well as the linear correlation of the response with each of the predictors, we start by checking the **correlation matrix**:

```
# We use the round function with 2 digits, to round the numbers in the output
# This only affects the printed numbers, not the ones R has stored.
round(cor(seatpos), dig=2)
```

```
##             Age Weight HtShoes    Ht Seated   Arm Thigh   Leg hipcenter
## Age        1.00   0.08   -0.08 -0.09  -0.17  0.36  0.09 -0.04      0.21
## Weight     0.08   1.00    0.83  0.83   0.78  0.70  0.57  0.78     -0.64
## HtShoes   -0.08   0.83    1.00  1.00   0.93  0.75  0.72  0.91     -0.80
## Ht        -0.09   0.83    1.00  1.00   0.93  0.75  0.73  0.91     -0.80
## Seated    -0.17   0.78    0.93  0.93   1.00  0.63  0.61  0.81     -0.73
## Arm        0.36   0.70    0.75  0.75   0.63  1.00  0.67  0.75     -0.59
## Thigh      0.09   0.57    0.72  0.73   0.61  0.67  1.00  0.65     -0.59
## Leg       -0.04   0.78    0.91  0.91   0.81  0.75  0.65  1.00     -0.79
## hipcenter  0.21  -0.64   -0.80 -0.80  -0.73 -0.59 -0.59 -0.79      1.00
```

We observe that the response is *highly correlated with most of the variables* which justifies the low p-value in the overall F test. However, we have **highly correlated predictors** (e.g. Corr(`Leg` , `HtShoes`)=0.91) which means that we will have *collinearity.*

We also calculate Variance Inflation Factor of model matrix X (after removing the first column) using function {vif(.)}.

```
# Extract the design matrix and remove the column of 1s that corresponds to the i
x = model.matrix(position.full)[,-1]

# Variance Inflation Factor (VIF)
round(vif(x), dig=2)
```

```
##     Age  Weight HtShoes      Ht  Seated     Arm   Thigh     Leg
##    2.00    3.65  307.43  333.14    8.95    4.50    2.76    6.69
```

Note that the *standard error* for the coefficient associated with `HtShoes` is **17.5 (=$\sqrt{307.43}$) times larger** than it would have been without collinearity.

A global measure of collinearity

A global measure of collinearity is given by examining the eigenvalues of $\mathbf{X}^\top\mathbf{X}$. A popular measure is the **condition number** of $\mathbf{X}^\mathbf{T}\mathbf{X}$, denoted by:

$$\kappa = (\text{largest eigenvalue/smallest eigenvalue})^{1/2}$$

An empirical rule for declaring collinearity is $\kappa \geq 30$. Note that κ is *not* scale-invariant, so we should standardize each column of X (i.e. each column should have zero mean and sample variance equal to 1), before calculating the condition number.

Car position Data Example

```
x = model.matrix(position.full)[,-1]

# Standardize the matrix
x = x - matrix(apply(x,2, mean), 38,8, byrow=TRUE)
x = x / matrix(apply(x, 2, sd), 38,8, byrow=TRUE)
```

Compute the condition number using the standardized matrix:

```
# Compute the eigenvalues of the matrix
eigenvalues.x = eigen(t(x) %*% x)
eigenvalues.x$val
```

```
## [1] 209.90786979  45.76108236  17.15850736   8.91545889   7.18612386
## [6]   5.14944541   1.86274750   0.05876483
```

```
# Compute the condition number:
sqrt(eigenvalues.x$val[1]/eigenvalues.x$val[8])
```

```
## [1] 59.7662
```

The condition number is **59.77**, larger than 30, so we conclude that *collinearity is present*.

To summarizy our findings, possible symptoms of collinearity include:

- high pair-wise (sample) correlation between predictors
- high VIF
- high condition number
- R^2 is relatively large but none of the predictor is significant.

When we find that collinearity is present, the easier remedial measure would be to remove the variables that are highly correlated to the rest. To do so, we study the *pairwise correlations* and perform *partial F-tests*.

Car position Data Example

```
cor(Seated+Thigh, Ht)
```

```
## [1] 0.9389819
```

```
cor(Seated+Leg, Ht)
```

```
## [1] 0.965607
```

```
cor(Seated+Arm, Ht)
```

```
## [1] 0.9465523
```

```
position.red1 = lm(hipcenter ~ Age + Weight + Ht + Seated, data=seatpos)
summary(position.red1)
```

```
##
## Call:
## lm(formula = hipcenter ~ Age + Weight + Ht + Seated, data = seatpos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -90.869 -21.163  -3.144  26.773  59.423
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 478.65890  159.73362   2.997  0.00515 **
## Age           0.58396    0.42573   1.372  0.17943
## Weight       -0.01535    0.31640  -0.049  0.96159
## Ht           -4.99025    1.64389  -3.036  0.00466 **
## Seated        2.04632    3.41283   0.600  0.55287
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 36.83 on 33 degrees of freedom
## Multiple R-squared:  0.6599, Adjusted R-squared:  0.6186
## F-statistic: 16.01 on 4 and 33 DF,  p-value: 2.224e-07
```

```r
position.red2 = lm(hipcenter ~ Ht, data=seatpos)
summary(position.red2)
```

```
##
## Call:
## lm(formula = hipcenter ~ Ht, data = seatpos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -99.956 -27.850   5.656  20.883  72.066
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 556.2553    90.6704   6.135 4.59e-07 ***
## Ht           -4.2650     0.5351  -7.970 1.83e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 36.37 on 36 degrees of freedom
## Multiple R-squared:  0.6383, Adjusted R-squared:  0.6282
## F-statistic: 63.53 on 1 and 36 DF,  p-value: 1.831e-09
```

```r
anova(position.red2, position.red1)
```

```
## Analysis of Variance Table
##
## Model 1: hipcenter ~ Ht
## Model 2: hipcenter ~ Age + Weight + Ht + Seated
##   Res.Df   RSS Df Sum of Sq      F Pr(>F)
## 1     36 47616
## 2     33 44774  3    2841.6 0.6981 0.5599
```

Based on the F test provided in the ANOVA table, we conclude that *the reduced model with `Ht` as the only variable is better than the model that includes `Age`, `Weight`, `Ht` and `Seated`.*