PE I: Multivariable Regression

Restricted Least Squares, Multicollinearity
(Chapters 4.4, 4.5)

Andrius Buteikis, andrius.buteikis@mif.vu.lt
http://web.vu.lt/mif/a.buteikis/

Multiple Regression: Model Assumptions

Much like in the case of the univariate regression with one independent variable, the multiple regression model has a number of required assumptions:

(MR.1): Linear Model The Data Generating Process (**DGP**), or in other words, the population, is described by a linear (*in terms of the coefficients*) model:

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon \tag{MR.1}$$

(MR.2): Strict Exogeneity Conditional expectation of ε, given all observations of the explanatory variable matrix $\mathbf{X}$, is zero:

$$\mathbb{E}(\varepsilon|\mathbf{X}) = \mathbf{0} \tag{MR.2}$$

This assumption also implies that $\mathbb{E}(\varepsilon) = \mathbb{E}(\mathbb{E}(\varepsilon|\mathbf{X})) = \mathbf{0}$, $\mathbb{E}(\varepsilon\mathbf{X}) = \mathbf{0}$ and $\mathbb{C}\text{ov}(\varepsilon, \mathbf{X}) = \mathbf{0}$. Furthermore, this property implies that: $\mathbb{E}(\mathbf{Y}|\mathbf{X}) = \mathbf{X}\beta$

(MR.3): Conditional Homoskedasticity The variance-covariance matrix of the error term, conditional on $\mathbf{X}$ is constant:

$$\mathbb{V}\text{ar}\left(\boldsymbol{\varepsilon}|\mathbf{X}\right) = \begin{bmatrix} \mathbb{V}\text{ar}(\epsilon_1) & \mathbb{C}\text{ov}(\epsilon_1, \epsilon_2) & \dots & \mathbb{C}\text{ov}(\epsilon_1, \epsilon_N) \\ \mathbb{C}\text{ov}(\epsilon_2, \epsilon_1) & \mathbb{V}\text{ar}(\epsilon_2) & \dots & \mathbb{C}\text{ov}(\epsilon_2, \epsilon_N) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{C}\text{ov}(\epsilon_N, \epsilon_1) & \mathbb{C}\text{ov}(\epsilon_N, \epsilon_2) & \dots & \mathbb{V}\text{ar}(\epsilon_N) \end{bmatrix} = \sigma_\epsilon^2 \mathbf{I} \qquad \text{(MR.3)}$$

(MR.4): Conditionally Uncorrelated Errors The covariance between different error term pairs, conditional on $\mathbf{X}$, is zero:

$$\mathbb{C}\text{ov}\left(\epsilon_i, \epsilon_j|\mathbf{X}\right) = 0, \quad i \neq j \qquad \text{(MR.4)}$$

This assumption implies that all error pairs are uncorrelated. For cross-sectional data, this assumption implies that there is no spatial correlation between errors.

(MR.5) There exists no exact linear relationship between the explanatory variables. This means that:

$$c_1 X_{i1} + c_2 X_{i2} + ... + c_k X_{ik} = 0, \ \forall i = 1, ..., N \iff c_1 = c_2 = ... = c_k = 0 \quad \text{(MR.5)}$$

This assumption is violated if there exists some $c_j \neq 0$.
Alternatively, this requirement means that:

$$\text{rank}\,(\mathbf{X}) = k + 1$$

or, alternatively, that:

$$\det\left(\mathbf{X}^\top \mathbf{X}\right) \neq 0$$

This assumption is important, because a linear relationship between independent variables means that we cannot separately estimate the effects of changes in each variable separately.

(MR.6) (optional) The residuals are normally distributed:

$$\varepsilon | \mathbf{X} \sim \mathcal{N}\left(\mathbf{0}, \sigma_\epsilon^2 \mathbf{I}\right) \quad \text{(MR.6)}$$

Restricted Least Squares

Suppose that we have the following multiple regression with k independent variables X_j, $j = 1, ..., k$, in matrix notation, with $M < k + 1$ different linear restrictions on the parameters:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$
$$\mathbf{L}\boldsymbol{\beta} = \boldsymbol{r}$$

where:

$$\mathbf{L} = \begin{bmatrix} c_{10} & c_{11} & \dots & c_{1k} \\ c_{20} & c_{21} & \dots & c_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ c_{M0} & c_{M1} & \dots & c_{Mk} \end{bmatrix}, \quad \boldsymbol{r} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_M \end{bmatrix}$$

Lagrange multipliers are widely used to solve various constrained optimization problems in economics.
In general, in order to find the **stationary points** of a function $f(\mathbf{X})$, subject to an **equality constraint** $g(\mathbf{X}) = \mathbf{0}$, from the **Lagrangian function**:

$$\mathcal{L}(\mathbf{X}, \boldsymbol{\lambda}) = f(\mathbf{X}) + \boldsymbol{\lambda} g(\mathbf{X})$$

Note that the solution corresponding to the above constrained optimization is always a saddle point of the Lagrangian function.

Estimator Derivation

As before, we want to minimize the sum of squares, but this time, they are subject to the condition that $\mathbf{L}\beta = \boldsymbol{r}$. This leads to the Lagrangian function:

$$\begin{aligned} L(\beta, \lambda) &= (\mathbf{Y} - \mathbf{X}\beta)^\top (\mathbf{Y} - \mathbf{X}\beta) + 2\lambda^\top (\mathbf{L}\beta - \boldsymbol{r}) \\ &= \mathbf{Y}^\top \mathbf{Y} - 2\mathbf{Y}^\top \mathbf{X}\beta + \beta^\top \mathbf{X}^\top \mathbf{X}\beta + 2\lambda^\top \mathbf{L}\beta - 2\lambda^\top \boldsymbol{r} \end{aligned}$$

Then equating the partial derivatives to zero:

$$\begin{cases} \dfrac{\partial L(\widehat{\beta}, \widehat{\lambda})}{\partial \widehat{\beta}} &= -2\mathbf{X}^\top \mathbf{Y} + 2\mathbf{X}^\top \mathbf{X}\widehat{\beta} + 2\lambda^\top \mathbf{L} = 0 \\ \dfrac{\partial L(\widehat{\beta}, \lambda)}{\partial \lambda} &= 2\mathbf{L}\beta - 2\boldsymbol{r} = 0 \end{cases}$$

After some rearranging, we have:

$$\begin{cases} \mathbf{X}^\top \mathbf{X}\widehat{\boldsymbol{\beta}} + \mathbf{L}^\top \boldsymbol{\lambda} = \mathbf{X}^\top \mathbf{Y} \\ \\ \mathbf{L}\boldsymbol{\beta} = \boldsymbol{r} \end{cases}$$

which we rewrite in the following block-matrix form:

$$\begin{bmatrix} \mathbf{X}^\top \mathbf{X} & \mathbf{L}^\top \\ \mathbf{L} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{\beta}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{X}^\top \mathbf{Y} \\ \boldsymbol{r} \end{bmatrix}$$

which we can further rewrite to get the solution:

$$\begin{bmatrix} \widehat{\boldsymbol{\beta}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{X}^\top \mathbf{X} & \mathbf{L}^\top \\ \mathbf{L} & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{X}^\top \mathbf{Y} \\ \boldsymbol{r} \end{bmatrix}$$

Alternatively, from $\frac{\partial L(\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{\lambda}})}{\partial \widehat{\boldsymbol{\beta}}} = 0$, we have that:

$$\widehat{\boldsymbol{\beta}}^{(RLS)} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{Y} - \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{L}^\top \boldsymbol{\lambda} = \widehat{\boldsymbol{\beta}}^{(OLS)} - \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{L}^\top \boldsymbol{\lambda}$$

Multiplying both sides by $\mathbf{L}$ yields:

$$\mathbf{L}\widehat{\boldsymbol{\beta}}^{(RLS)} = \mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)} - \mathbf{L}\left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{L}^\top \boldsymbol{\lambda}$$

Since $\mathbf{L}\widehat{\boldsymbol{\beta}}^{(RLS)} = \boldsymbol{r}$, we can express $\boldsymbol{\lambda}$ as:

$$\boldsymbol{\lambda} = \left(\mathbf{L}\left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{L}^\top\right)^{-1} \left(\mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)} - \boldsymbol{r}\right)$$

Then, we can substitute it into the initial $\widehat{\boldsymbol{\beta}}^{(RLS)}$ expression to get:

$$\widehat{\boldsymbol{\beta}}^{(RLS)} = \widehat{\boldsymbol{\beta}}^{(OLS)} - \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{L}^\top \left(\mathbf{L}\left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{L}^\top\right)^{-1} \left(\mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)} - \boldsymbol{r}\right)$$

This means is that the RLS estimator can be defined as:

$$\widehat{\boldsymbol{\beta}}^{(RLS)} = \widehat{\boldsymbol{\beta}}^{(OLS)} + \text{"Restriction Adjustment"}$$

where the `Restriction Adjustment` is the divergence between $\mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)}$ and $\mathbb{E}\left(\mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)}\right) = \boldsymbol{r}$.

We note the following terms can be expressed as:

$$\mathbb{V}\mathrm{ar}\left(\mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)}\right) = \mathbf{L}\mathbb{V}\mathrm{ar}\left(\widehat{\boldsymbol{\beta}}^{(OLS)}\right)\mathbf{L}^{\top} = \sigma^2\mathbf{L}\left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1}\mathbf{L}^{\top}$$

$$\mathbb{C}\mathrm{ov}\left(\widehat{\boldsymbol{\beta}}^{(OLS)},\ \mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)}\right) = \sigma^2\left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1}\mathbf{L}^{\top}$$

which means that the RLS estimator is:

$$\widehat{\boldsymbol{\beta}}^{(RLS)} = \widehat{\boldsymbol{\beta}}^{(OLS)} - \mathbb{C}\mathrm{ov}\left(\widehat{\boldsymbol{\beta}}^{(OLS)},\ \mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)}\right)\left(\mathbb{V}\mathrm{ar}\left(\mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)}\right)\right)^{-1}\left(\mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)} - \boldsymbol{r}\right)$$

Properties of the RLS Estimator

Similarly to OLS, the RLS estimator has a number of properties, which can be derived and verified.

Exact Restrictions with RLS Estimator

To verify that $\mathbf{L}\widehat{\boldsymbol{\beta}}^{(RLS)} = \mathbf{r}$, multiply the expression of $\widehat{\boldsymbol{\beta}}^{(RLS)}$ by $\mathbf{L}$ to get:

$$\begin{aligned}
\mathbf{L}\widehat{\boldsymbol{\beta}}^{(RLS)} &= \mathbf{L}\left[\widehat{\boldsymbol{\beta}}^{(OLS)} - \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{L}^\top\left(\mathbf{L}\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{L}^\top\right)^{-1}\left(\mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)} - \boldsymbol{r}\right)\right] \\
&= \mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)} - \left(\mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)} - \boldsymbol{r}\right) \\
&= \boldsymbol{r}
\end{aligned}$$

Unbiasedness of RLS Estimator

We begin by examining the estimation error of estimating $\boldsymbol{\beta}$ via RLS in a similar way as we did for OLS:

$$\begin{aligned}\widehat{\boldsymbol{\beta}}^{(RLS)} &= \widehat{\boldsymbol{\beta}}^{(OLS)} - \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{L}^\top\left(\mathbf{L}\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{L}^\top\right)^{-1}\left(\mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)} - \mathbf{L}\boldsymbol{\beta}\right)\\ &= \widehat{\boldsymbol{\beta}}^{(OLS)} + \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{L}^\top\left(\mathbf{L}\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{L}^\top\right)^{-1}\mathbf{L}\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}^{(OLS)}\right)\end{aligned}$$

if $\mathbf{L}\boldsymbol{\beta} = \boldsymbol{r}$.

Then the expected value of the RLS estimator is:

$$\mathbb{E}\left(\widehat{\boldsymbol{\beta}}^{(RLS)}\right) = \mathbb{E}\left(\widehat{\boldsymbol{\beta}}^{(OLS)}\right) + \left[\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{L}^\top\left(\mathbf{L}\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{L}^\top\right)^{-1}\mathbf{L}\right]\mathbb{E}\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}^{(OLS)}\right) = \boldsymbol{\beta}$$

if $\mathbb{E}\left(\widehat{\boldsymbol{\beta}}^{(OLS)}\right) = \boldsymbol{\beta}$.

In other words:

> If the **OLS** estimator is **unbiased**, then the RLS estimator is also unbiased as long as the constraints $\mathbf{L}\boldsymbol{\beta} = \boldsymbol{r}$ are true.
>
> Otherwise, the RLS estimator is **biased**, i.e. $\mathbb{E}\left(\widehat{\boldsymbol{\beta}}^{(RLS)}\right) \neq \boldsymbol{\beta}$.

Efficiency of RLS Estimator

Noting that the difference $\widehat{\boldsymbol{\beta}}^{(RLS)} - \boldsymbol{\beta}$ can be expressed as:

$$\begin{aligned}
\widehat{\boldsymbol{\beta}}^{(RLS)} - \boldsymbol{\beta} &= \widehat{\boldsymbol{\beta}}^{(OLS)} - \boldsymbol{\beta} - \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{L}^\top\left(\mathbf{L}\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{L}^\top\right)^{-1}\left(\mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)} - \mathbf{L}\boldsymbol{\beta}\right) \\
&= \left[\mathbf{I} - \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{L}^\top\left(\mathbf{L}\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{L}^\top\right)^{-1}\mathbf{L}\right]\left(\widehat{\boldsymbol{\beta}}^{(OLS)} - \boldsymbol{\beta}\right) \\
&= \mathbf{D}\left(\widehat{\boldsymbol{\beta}}^{(OLS)} - \boldsymbol{\beta}\right)
\end{aligned}$$

where $\mathbf{D} = \mathbf{I} - \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{L}^\top\left(\mathbf{L}\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{L}^\top\right)^{-1}\mathbf{L}$.

This allows us to calculate the variance of the RLS estimator:

$$\begin{aligned}\mathbb{V}\mathrm{ar}\left(\widehat{\boldsymbol{\beta}}^{(RLS)}\right) &= \mathbb{E}\left[(\widehat{\boldsymbol{\beta}}^{(RLS)}-\boldsymbol{\beta})(\widehat{\boldsymbol{\beta}}^{(RLS)}-\boldsymbol{\beta}))^{\top}\right] = \mathbf{D}\mathbb{E}\left[(\widehat{\boldsymbol{\beta}}^{(RLS)}-\boldsymbol{\beta})(\widehat{\boldsymbol{\beta}}^{(RLS)}-\boldsymbol{\beta})^{\top}\right]\mathbf{D}^{\top} \\ &= \sigma^2\mathbf{D}\left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1}\mathbf{D}^{\top} = \sigma^2\mathbf{D}\left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1} \\ &= \sigma^2\left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1} - \sigma^2\left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1}\mathbf{L}^{\top}\left(\mathbf{L}\left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1}\mathbf{L}^{\top}\right)^{-1}\mathbf{L}\left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1}\end{aligned}$$

Since $\mathbb{V}\mathrm{ar}\left(\widehat{\boldsymbol{\beta}}^{(OLS)}\right) = \sigma^2\left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1}$, we can calculate the difference in the variances of the RLS and OLS estimators:

$$\mathbb{V}\mathrm{ar}\left(\widehat{\boldsymbol{\beta}}^{(OLS)}\right) - \mathbb{V}\mathrm{ar}\left(\widehat{\boldsymbol{\beta}}^{(RLS)}\right) = \sigma^2\left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1}\mathbf{L}^{\top}\left(\mathbf{L}\left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1}\mathbf{L}^{\top}\right)^{-1}\mathbf{L}\left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1} = \sigma^2\mathbf{C}$$

It can be shown that the matrix $\mathbf{C}$ is a **positive semi-definite** matrix. Consequently, the diagonal elements of $\mathrm{diag}\left(\mathbb{V}\mathrm{ar}\left(\widehat{\boldsymbol{\beta}}^{(RLS)}\right)\right) \leq \mathrm{diag}\left(\mathbb{V}\mathrm{ar}\left(\widehat{\boldsymbol{\beta}}^{(OLS)}\right)\right)$.

This means that:

> If the a priori information on the parameters is correct (i.e. if the restrictions $\mathbf{L}\boldsymbol{\beta} = \boldsymbol{r}$ are true), the RLS estimator is **more efficient** than the OLS estimator.
> It can be shown that even if $\mathbf{L}\boldsymbol{\beta} \neq \boldsymbol{r}$, then the RLS estimator remains at least as efficient as the OLS estimator.

Consistency of RLS Estimator

From the OLS estimator we have that $\widehat{\boldsymbol{\beta}}^{(OLS)} \to \boldsymbol{\beta}$. This means that $\left(\widehat{\boldsymbol{\beta}}^{(OLS)} - \boldsymbol{\beta}\right) \to \mathbf{0}$.

Applying this to the RLS estimator yields:

$$\widehat{\boldsymbol{\beta}}^{(RLS)} = \widehat{\boldsymbol{\beta}}^{(OLS)} + \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{L}^\top\left(\mathbf{L}\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{L}^\top\right)^{-1}\left(\boldsymbol{r} - \mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)}\right) \to \boldsymbol{\beta}$$

if $\mathbf{L}\boldsymbol{\beta} = \boldsymbol{r}$ (which we can then plug in our RLS estimator expression). So, the RLS estimator is a **consistent** estimator of $\boldsymbol{\beta}$, provided the restrictions are correctly specified.

Estimating the Error Variance

According to the RLS estimator, the residual vector can be defined as:

$$\widehat{\varepsilon} = \mathbf{Y} - \mathbf{X}\widehat{\beta}_{RLS}$$

The error variance is derived analogously to how it would be via OLS, except we have M restrictions, so now:

The RLS (with M restrictions) estimator of the error variance σ^2:

$$\widehat{\sigma}^2_{RLS} = \frac{\widehat{\varepsilon}^\top \widehat{\varepsilon}}{N - (k + 1 - M)}$$

is an unbiased estimator of $\widehat{\sigma}^2$.

Some final notes about the RLS estimator:

If the restrictions are **correct**, $\mathbf{L}\boldsymbol{\beta} = \boldsymbol{r}$, then:

- $\widehat{\boldsymbol{\beta}}^{(RLS)}$ is **unbiased**;
- $\widehat{\boldsymbol{\beta}}^{(RLS)}$ is **more efficient** than $\widehat{\boldsymbol{\beta}}^{(OLS)}$;
- $\widehat{\boldsymbol{\beta}}^{(RLS)}$ is **BLUE**;
- $\widehat{\boldsymbol{\beta}}^{(RLS)}$ is **consistent**;
- $\widehat{\boldsymbol{\beta}}^{(RLS)}$ is asymptotically normally distributed:

$$\widehat{\boldsymbol{\beta}}^{(RLS)} \sim \mathcal{N}\left(\boldsymbol{\beta},\ \sigma^2 \mathbf{D}\left(\mathbf{X}^\top \mathbf{X}\right)^{-1}\right)$$

where $\mathbf{D} = \mathbf{I} - \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{L}^\top \left(\mathbf{L}\left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{L}^\top\right)^{-1} \mathbf{L}$.

If the restrictions are **incorrect**, $\mathbf{L}\boldsymbol{\beta} \neq \boldsymbol{r}$, then:

- $\widehat{\boldsymbol{\beta}}^{(RLS)}$ is **biased** (and thus, **no longer BLUE**);
- $\widehat{\boldsymbol{\beta}}^{(RLS)}$ is **remains more efficient** than $\widehat{\boldsymbol{\beta}}^{(OLS)}$;
- $\widehat{\boldsymbol{\beta}}^{(RLS)}$ is **inconsistent**;

So, the variance of $\widehat{\boldsymbol{\beta}}^{(RLS)}$ is smaller than $\widehat{\boldsymbol{\beta}}^{(OLS)}$, whether the constraints are true or not.

Testing for Linear Restrictions

In practice, we begin by estimating an **unrestricted** model via **OLS**. We then test the following hypothesis of M linear restrictions:

$$\begin{cases} H_0 & : \mathbf{L}\boldsymbol{\beta} = \boldsymbol{r} \\ H_1 & : \mathbf{L}\boldsymbol{\beta} \neq \boldsymbol{r} \end{cases}$$

We can test it with the F-test, presented in subsection @ref(Multiple-F-Test-Restrict):

$$F = \frac{1}{M}\left(\mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)} - \boldsymbol{r}\right)^{\top}\left[\mathbf{L}\widehat{\mathbb{V}\mathrm{ar}}\left(\widehat{\boldsymbol{\beta}}^{(OLS)}\right)\mathbf{L}^{\top}\right]^{-1}\left(\mathbf{L}\widehat{\boldsymbol{\beta}}^{(OLS)} - \boldsymbol{r}\right) \sim F_{(M,N-(k+1))}$$

Then:

- If the null hypothesis cannot be rejected - re-estimate the model via RLS;
- If the null hypothesis is rejected - keep the OLS estimates.

As we have seen - RLS is biased, unless the imposed constraints are **exactly** true.
Consequently, being knowledgeable about economics allows one to:

- Identify which variables may effect the dependent variable;
- Specify the model in a correct functional form;
- **Specify any additional restrictions on the coefficients**.

When these restrictions, as well as the model form itself, *primarily* come from your knowledge of economic theory, rather than from the data sample, they may be regarded as **nonsample information**.

As we have seen thus far, there are many different combinations of variables and their transformations, which would lead to a near-infinite number of possible models.

This is where economic theory is very useful.

- As we have noted, evidence of whether the imposed restrictions are true or not can be obtained by carrying out the specified F-test.
- Then again, if the test rejects the hypothesis about our constraints, but we are absolutely sure that they are valid from economic theory, then our RLS estimation may be closer to the true model (it will have lower variance), but the hypothesis may be rejected due to other reasons (for example, an incorrect model form, or omitted important variable).

If we are wrong, then the RLS estimates will be biased (though their variance will still be lower than OLS). In general, biased estimates are undesirable.

Example

We will simulate the following model:

$$Y_i = \beta_0 + \beta_1 X_{1,i} + \beta_2 X_{2,i} + \beta_3 X_{3,i} + \beta_4 X_{4,i} + \beta_5 X_{5,i} + \epsilon_i$$

where:

- $\beta_1 = \beta_3$;
- $(-2) \cdot \beta_2 = \beta_4$;
- $\beta_5 = 0$ (i.e. insignificant variable);

```
set.seed(123)
#
N <- 1000
beta_vec <- c(-5, 2, 3, 2, -6, 0)
#
x1 <- rnorm(mean = 5, sd = 2, n = N)
x2 <- sample(1:50, size = N, replace = TRUE)
x3 <- seq(from = 0, to = 10, length.out = N)
x4 <- rnorm(mean = 10, sd = 3, n = N)
x5 <- rnorm(mean = -2, sd = 3, n = N)
#
x_mat <- cbind(x1, x2, x3, x4, x5)
e <- rnorm(mean = 0, sd = 3, n = N)
y <- cbind(1, x_mat) %*% beta_vec + e
data_smpl <- data.frame(y, x_mat)
```

Next up, we will estimate the relevant model:

```
y_mdl_fit <- lm("y ~ x1 + x2 + x3 + x4 + x5", data = data_smpl)
#
print(round(coef(summary(y_mdl_fit)), 5))
```

```
##             Estimate Std. Error    t value Pr(>|t|)
## (Intercept) -5.49252    0.47593  -11.54062  0.00000
## x1           2.01919    0.04760   42.42357  0.00000
## x2           3.01065    0.00658  457.86812  0.00000
## x3           2.02373    0.03265   61.98141  0.00000
## x4          -6.00084    0.03180 -188.71161  0.00000
## x5           0.01270    0.03121    0.40704  0.68407
```

We can see that the coefficients (and their significance) is similar to what we used to generate the data sample.

Next up, we will carry out an F-test to test the following hypothesis:

$$\begin{cases} H_0 & : \beta_1 = \beta_3 \text{ and } (-2) \cdot \beta_2 = \beta_4 \text{ and } \beta_5 = 0 \\ H_1 & : \text{at least one equality doesn't hold} \end{cases}$$

Note that we can re-write the hypothesis alternatively as:

$$\begin{cases} H_0 & : \beta_1 - \beta_3 = 0 \text{ and } 2 \cdot \beta_2 + \beta_4 = 0 \text{ and } \beta_5 = 0 \\ H_1 & : \text{at least one equality doesn't hold} \end{cases}$$

```r
car::linearHypothesis(y_mdl_fit, c("x1 - x3 = 0", "2*x2 + x4 = 0", "x5 = 0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## x1 - x3 = 0
## 2 x2  + x4 = 0
## x5 = 0
##
## Model 1: restricted model
## Model 2: y ~ x1 + x2 + x3 + x4 + x5
##
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    997 8845.0
## 2    994 8840.1  3    4.9046 0.1838 0.9074
```

which is equivalent to using a matrix restriction specification:

```
L <- matrix(c(0, 1, 0, -1, 0, 0,
              0, 0, 2, 0, 1, 0,
              0, 0, 0, 0, 0, 1), nrow = 3, byrow = TRUE)
rr <- c(0, 0, 0)
car::linearHypothesis(y_mdl_fit, hypothesis.matrix = L, rhs = rr)
```

```
## Linear hypothesis test
##
## Hypothesis:
## x1 - x3 = 0
## 2 x2  + x4 = 0
## x5 = 0
##
## Model 1: restricted model
## Model 2: y ~ x1 + x2 + x3 + x4 + x5
##
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    997 8845.0
## 2    994 8840.1  3    4.9046 0.1838 0.9074
```

The p-value > 0.05, so we do not reject the null hypothesis.

Since we do not reject the null hypothesis - RLS would be more efficient than OLS and still be BLUE and Consistent.

Let's move on to estimating the regression via RLS. We can extract the formula used in the `lm()` function via the `formula()` function. In order to estimate the model via RLS, we need to use the `rls` function from the `lrmest` package:

```
formula(y_mdl_fit)
```

```
## y ~ x1 + x2 + x3 + x4 + x5
## <environment: 0x000000001b910ea8>
```

```
L <- matrix(c(0, 1, 0, -1, 0, 0,
              0, 0, 2, 0, 1, 0,
              0, 0, 0, 0, 0, 1), nrow = 3, byrow = TRUE)
rr <- c(0, 0, 0)
#
mdl_rls = lrmest::rls(formula = formula(y_mdl_fit), R = L, r = rr, data = data_smpl, delt = rep(0, le
```

```
## Warning in model.matrix.default(md, cal, contrasts): non-list contrasts
## argument ignored
```

```
print(mdl_rls[[1]])
```

```
##             Estimate Standard_error t_statistic pvalue
## (Intercept)  -5.3174         0.2885           0      1
## x1            2.0226         0.0269           0      1
## x2            3.0091         0.0061           0      1
## x3            2.0226         0.0269          NA     NA
## x4           -6.0181         0.0122          NA     NA
## x5            0.0000         0.0000          NA     NA
```

Note: The warning can most likely be safely ignored, as discussed here

Manual Calculation

As always, we can estimate the coefficients manually. This is especially useful when the output contains `warnings`, `errors`, `NA`, or `nan` values and we are not sure, whether we trust the output. Looking at the RLS formula, we will re-specify all of the required coefficients in one place, so as to make it easier to understand:

```
L <- matrix(c(0, 1, 0, -1, 0, 0,
              0, 0, 2, 0, 1, 0,
              0, 0, 0, 0, 0, 1), nrow = 3, byrow = TRUE)
rr <- c(0, 0, 0)
# Design matrix
X_d <- cbind(1, x_mat)
```

Next, we need to estimate the parameters via OLS:

```
xtx_inv  <- solve(t(X_d) %*% X_d)
beta_ols <- xtx_inv %*% t(X_d) %*% y
#
print(beta_ols)
```

```
##            [,1]
##    -5.49251716
## x1  2.01919172
## x2  3.01064730
## x3  2.02373361
## x4 -6.00084216
## x5  0.01270398
```

Then, since the RLS estimator can be expressed as $\widehat{\beta}^{(RLS)} = \widehat{\beta}^{(OLS)} +$ "Restriction Adjustment", we will calculate the adjustment component separately:

```
RA_1 <- xtx_inv %*% t(L)
RA_2 <- solve(L %*% xtx_inv %*% t(L))
RA_3 <- L %*% beta_ols - rr
RA <- RA_1 %*% RA_2 %*% RA_3
```

Having calculated the adjustment component, we can finally calculate the **RLS** estimator:

```
beta_rls <- beta_ols - RA
print(beta_rls)
```

```
##          [,1]
##    -5.317401
## x1  2.022635
## x2  3.009071
## x3  2.022635
## x4 -6.018142
## x5  0.000000
```

Since we know the asymptotic distribution, we can calculate the variance of the estimates:

```
y_fit <- X_d %*% beta_rls
#
resid <- y - y_fit
#
sigma2_rls <- sum(resid^2) / (nrow(data_smpl) - (length(beta_rls) - length(rr)))
print(paste0("Estimated RLS residual standard error: ", round(sqrt(sigma2_rls), 5)))
```

```
## [1] "Estimated RLS residual standard error: 2.97853"
```

Next, we will calculate the matrix **D**, which we will need to estimate the RLS standard errors. Fortunately, we can use some of the partial expressions that we have calculated for the adjustment component in RLS estimator calculation:

```
D_mat <- diag(length(beta_rls)) - RA_1 %*% RA_2 %*% L
rls_vcov <- sigma2_rls * D_mat %*% xtx_inv
#
beta_rls_se <- sqrt(diag(rls_vcov))
```

```
print(data.frame(est = beta_rls,
                 se = beta_rls_se))
```

```
##          est           se
##    -5.317401 2.881625e-01
## x1  2.022635 2.687947e-02
## x2  3.009071 6.113634e-03
## x3  2.022635 2.687947e-02
## x4 -6.018142 1.222727e-02
## x5  0.000000 1.851409e-11
```

Note the zero-value restriction on the last coefficient. Because of that restriction the variance of x5 coefficient is very close to zero (as it should be) but it is can also sometimes be **negative**:

```
print(tail(diag(rls_vcov), 1))
```

```
##           x5
## 3.427715e-22
```

- For cases where we have an insignificant parameter, instead of imposing a zero-value separate restriction in the RLS, it would be better to remove it first, and then re-estimate the model via RLS for any other restrictions (like parameter equality, difference, sum, ratio and other linear restrictions).
- We would like to stress that, knowing how the estimates are calculated is very useful, since it is possible to identify the source of any errors, or warnings, that we may get using parameter estimator functions from various packages, which may vary in functionality, speed and quality.

Multicollinearity

Definition of Multicollinearity

- Sometimes explanatory variables are tightly connected and it is impossible to distinguish their **individual** influences on the dependent variable.
- Many economic variables may move together in some systematic way. Such variables are said to be **collinear** and cause the **collinearity** problem.
- In the same way, **multicollinearity** refers to a situation in which **two or more** explanatory variables are highly linearly related.

A set of variables is perfectly multicollinear if there exist **one of more exact linear relationship** among some of the variables. This means that if for a multiple regression model:

$$Y_i = \beta_0 + \beta_1 X_{1,i} + ... + \beta_k X_{k,i} + \epsilon_i$$

the following relationship holds true for m explanatory variables:

$$\gamma_0 + \gamma_1 X_{1,i} + ... + \gamma_k X_{k,i} = 0, \forall i = 1, ..., N$$

for **some** $\gamma_j \neq 0$, $j = 1, ..., k$. Then these m variables exhibit **multicollinearity** and **violate assumption (MR.5)**.

Example

Assume that all advertising types (newspaper ads, website ads, etc.) are **coordinated**, i.e. we change them all at once (and cannot do so individually). Then, it may be difficult to distinguish how different forms of advertising affect revenue.

While it is clear that total advertising expenditure increases sales revenue, it is difficult to identify the individual effect of each advertising type, since it is impossible for either type to remain unchanged, if any other type changes (**this is from our assumption**), and as such, we never observed such cases.

On the other hand, **if we know for a fact** that there is no coordination between different advertisement types (e.g. they can independently increase or decrease), then we would be able to successfully identify their individual effects.

Example

Assume that we are analysing a production of some product.
The production (function) takes various inputs in various **proportions**: labor, capital, raw materials, electricity, etc.
As production increases, changing any one variable (e.g. increase in labor) causes **proportional** changes in other variables.
Consequently, any effort to measure the individual effects (marginal products) of various inputs from such data will be difficult.

Example

Consider the following model:

$$Y_i = \beta_0 + \beta_1 X_{1,i} + \beta_2 X_{2,i} + \epsilon_i$$

Now, assume that $X_{2,i} = \gamma_0 + \gamma_1 X_{1,i}$. Then our initial expression becomes:

$$\begin{aligned} Y_i &= \beta_0 + \beta_1 X_{1,i} + \beta_2(\gamma_0 + \gamma_1 X_{2,i}) + \epsilon_i \\ &= [\beta_0 + \beta_2\gamma_0] + [\beta_1 + \beta_2\gamma_1] \cdot X_{1,i} + \epsilon \\ &= \alpha_0 + \alpha_1 X_{1,i} + \epsilon \end{aligned}$$

Thus, while we may be able to estimate α_0 and α_1, we would not be able to obtain estimates of the original β_0, β_1, β_2.

To verify that we cannot estimate the original coefficients, consider the following example.

Example

We will generate a simple example of two models of the following form:

$$Y_{1,i} = \beta_0 + \beta_1 X_{1,i} + \beta_2 X_{2,i} + \epsilon_i$$
$$Y_{2,i} = \beta_0 + \beta_1 X_{1,i} + \beta_2 X_{3,i} + \epsilon_i$$

where $X_{1,i}$ and $X_{2,i}$ are **independent**, but $X_{3,i} = \gamma_0 + \gamma_1 X_{1,i}$.

```
set.seed(123)
#
N = 200
beta_vec <- c(2, 4, -2)
x1 = rnorm(mean = 8, sd = 2, n = N)
x2 = rnorm(mean = 2, sd = 1, n = N)
x3 = 2 - 2 * x1
e  = rnorm(mean = 0, sd = 1, n = N)
#
y1 = cbind(1, x1, x2) %*% beta_vec + e
y2 = cbind(1, x1, x3) %*% beta_vec + e
```

If we were to manually examine the design matrices of our models, we can verify that **assumption (MR.5) does not hold** for the design matrix of the second model:

```
x_mat_1 <- cbind(1, x1, x2)
print(Matrix::rankMatrix(x_mat_1)[1])
```

```
## [1] 3
```

```
x_mat_2 <- cbind(1, x1, x3)
print(Matrix::rankMatrix(x_mat_2)[1])
```

```
## [1] 2
```

Consequently, calculating the inverse of $\mathbf{X}^\top \mathbf{X}$ yields an error:

```
solve(t(x_mat_2) %*% x_mat_2)
```

```
## Error in solve.default(t(x_mat_2) %*% x_mat_2): system is computationally singular: reciprocal cor
```

Strangely, this works in Python:

```
import numpy as np
import statsmodels.api as sm
#
np.random.seed(123)
#
N = 200
beta_vec = np.array([2, 4, -2])
x1 = np.random.normal(loc = 8, scale = 2, size = N)
x2 = np.random.normal(loc = 2, scale = 1, size = N)
x3 = 2 - 2 * x1
e  = np.random.normal(loc = 0, scale = 1, size = N)
#
y1 = np.dot(sm.add_constant(np.column_stack((x1, x2))), beta_vec) + e
y2 = np.dot(sm.add_constant(np.column_stack((x1, x3))), beta_vec) + e
```

```
x_mat_1 = sm.add_constant(np.column_stack((x1, x2)))
print(np.linalg.matrix_rank(x_mat_1))
```

```
## 3
```

```
x_mat_2 = sm.add_constant(np.column_stack((x1, x3)))
print(np.linalg.matrix_rank(x_mat_2))
```

```
## 2
```

```
print(np.linalg.inv(np.dot(np.transpose(x_mat_2), x_mat_2)))
```

```
## [[ 7.33007752e+11 -7.33007752e+11 -3.66503876e+11]
##  [-7.33007752e+11  7.33007752e+11  3.66503876e+11]
##  [-3.66503876e+11  3.66503876e+11  1.83251938e+11]]
```

It turns out that `numpy` uses LU decomposition to calculate the inverse., which yields results even in cases when the matrix is singular (i.e. non-invertible), we can check this not only by examining its rank, but from the determinant:

```
print(np.linalg.det(np.dot(np.transpose(x_mat_2), x_mat_2)))
```

```
## 9.669827929485215e-07
```

In theory, if we have calculated the inverse matrix, then the following relation should hold:

$$\left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \left(\mathbf{X}^\top \mathbf{X}\right) = \mathbf{I}$$

For the **invertible matrix** this holds true (as it should):

```
xtx_1 = np.dot(np.transpose(x_mat_1), x_mat_1)
xtx_1_inv = np.linalg.inv(xtx_1)
print(np.round(np.dot(xtx_1_inv, xtx_1), 10))
```

```
## [[ 1. -0. -0.]
##  [-0.  1. -0.]
##  [-0.  0.  1.]]
```

On the other hand, for the **non-invertible matrix**, this does not hold (because the inverse **should not exist**):

```
xtx_2 = np.dot(np.transpose(x_mat_2), x_mat_2)
xtx_2_inv = np.linalg.inv(xtx_2)
print(np.round(np.dot(xtx_2_inv, xtx_2), 10))
```

```
## [[  1.51922246   5.13709534 -10.90241243]
##  [  0.01652794   0.32784124   1.79404007]
##  [  0.034764     0.03262621   1.21260891]]
```

It should be pointed out that this is extremely dependent on your package version as discussed here.

As of writing these lecture notes, this remains a "problem" (in the sense that you need to be mindful when doing these calculations in `Python`) with the specific package versions (see the software section in the lecture notes for the versions used here).

We note that, again, when there is perfect collinearity, the design matrix should **not** be invertible, as is clearly demonstrated in `R`.

On the other hand, **dropping** the variable $X_{3,i}$ (and thus, losing some potentially valuable information on our dependent variable) yields the following coefficients:

```
x_mat_2_small <- cbind(1, x1)
print(solve(t(x_mat_2_small) %*% x_mat_2_small) %*% t(x_mat_2_small) %*% y2)
```

```
##         [,1]
##    -1.844784
## x1  7.984537
```

Consequently, this is exactly what is (**implicitly**) done in R. If we estimate the models separately, we get:

```
dt1 <- data.frame(y1, x1, x2)
dt2 <- data.frame(y2, x1, x3)
#
mdl_1 <- lm(y1 ~ x1 + x2, data = dt1)
print(coef(summary(mdl_1)))
```

```
##              Estimate Std. Error    t value      Pr(>|t|)
## (Intercept)  2.264582 0.33337750   6.792846  1.269838e-10
## x1           3.983796 0.03638855 109.479387 2.073561e-178
## x2          -2.050659 0.06892157 -29.753510  8.319799e-75
```

```
mdl_2 <- lm(y2 ~ x1 + x3, data = dt2)
print(coef(summary(mdl_2)))
```

```
##              Estimate Std. Error   t value      Pr(>|t|)
## (Intercept) -1.844784 0.29798365  -6.19089  3.384508e-09
## x1           7.984537 0.03633232 219.76400 1.739763e-238
```

Furthermore, if we print the summary of our model with collinear variables:

```
print(summary(mdl_2))
```

```
##
## Call:
## lm(formula = y2 ~ x1 + x3, data = dt2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.83465 -0.59224  0.06078  0.64298  2.46366
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.84478    0.29798  -6.191 3.38e-09 ***
## x1           7.98454    0.03633 219.764  < 2e-16 ***
## x3                NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9668 on 198 degrees of freedom
## Multiple R-squared:  0.9959, Adjusted R-squared:  0.9959
## F-statistic: 4.83e+04 on 1 and 198 DF,  p-value: < 2.2e-16
```

We see that the variable X_3 has `NA` values - in other words, it is not included in our `lm()` estimated regression which is why `coef()` does not return the row with `x3` variable.

- Always be mindful of the possible **consequences** of collinearity (which results in an non-invertible matrices in OLS calculation), as well as other possible problems like autocorrelation and heteroskedasticity (which are to be discussed further on).
- Econometric software is not always explicit in its methodology when some of these problems arise - it is usually assumed that the user if *knowledgeable enough* to identify these issues from the *output*, or **by reading the software documentation**.
- We may sometimes be content with an approximate solution, while other times, we want to be fully aware of any and all possible problems with the data, **especially if we are automating some model estimation processes**, since extracting only the coefficient (or any other) values may not tell us about any problems *detected by the software*.

Consequences of Multicollinearity

While we may be able to estimate γ_0 and γ_1, we would not be able to obtain estimates of the original β_0, β_1, β_2.

- On one hand, **this situation virtually never arises in practice and can be disregarded**.
- On the other hand, the relationship in real-world data is usually **approximate**.

In other words if the relationship is approximately linear:

$$X_{3,i} \approx \gamma_0 + \gamma_1 f(X_{1,i}) + \eta_i, \quad \eta_i \sim D(\mu_\eta, \sigma_\eta^2)$$

where η is some kind of (not necessarily Gaussian) random variable. If η is small, then $\det\left(\mathbf{X}^\top \mathbf{X}\right)$ is *close* to zero, then our data is close to multicollinear and $\left(\mathbf{X}^\top \mathbf{X}\right)^{-1}$ would be *large*, which would then result in *imprecise* estimators.

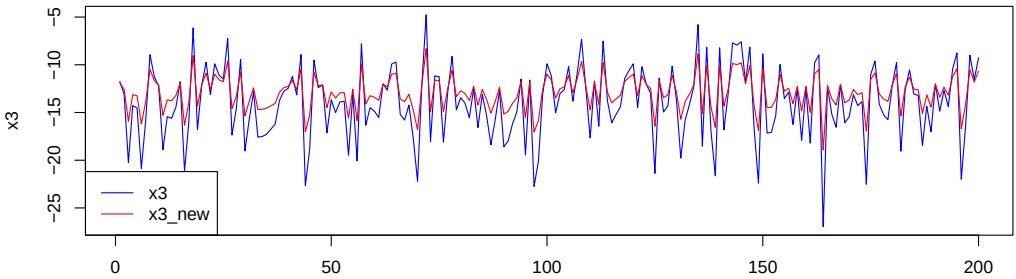
Consequently multicollinearity results in multiple difficulties:

- **That the standard errors of the affected coefficients tend to be large.** This results in failure to reject the null hypothesis on coefficient significance.
- **Small changes to the input data can lead to large changes in the model, even resulting in changes of sign of parameter estimates.** This is easier to verify if we estimate the same model on different subsets of the data and examine its coefficients.
- R^2 may be strangely large, which should be concerning when the model contains insignificant variables and variables with (economically) incorrect signs.
- The usual interpretation of a unit increase in $X_{j,i}$ effect on Y, **holding all else constant** (ceteris paribus) **does not hold if the explanatory variables are correlated**.
- In some sense, the collinear variables contain, at least partially, the same information about the dependent variable. A consequence of such data redundancy is **overfitting**.
- Depending on the software used, an *approximate inverse* of $\mathbf{X}^\top\mathbf{X}$ may be computed, although the resulting approximated inverse may be highly sensitive to slight variations in the data and so may be very inaccurate or very sample-dependent.

Modifying our previous example:

$$X_{3,i} = 2 - 2 \cdot (X_{1,i})^{0.9} + \eta_i, \quad \eta_i \sim \mathcal{N}(-2, 1)$$

```r
set.seed(123)
#
x3_new = 2 - 2 * x1^(0.9) + rnorm(mean = -2, sd = 1, n = N)
y2_new = cbind(1, x1, x3_new) %*% beta_vec + e
#
plot(1:N, x3, type = "l", col = "blue")
lines(1:N, x3_new, col = "red")
legend("bottomleft", legend = c("x3", "x3_new"), col = c("blue", "red"), lty = 1)
```



```
print(cor(cbind(x3, x3_new)))
```

```
##                x3    x3_new
## x3     1.0000000 0.9996801
## x3_new 0.9996801 1.0000000
```

yields the following model coefficient estimates:

```
dt3 <- data.frame(y2_new, x1, x3_new)
#
mdl_3 <- lm(y2_new ~ x1 + x3_new, data = dt3)
#
print(round(coef(summary(mdl_3)), 4))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.2029     7.9048  0.9112   0.3633
## x1            4.9036     1.4387  3.4084   0.0008
## x3_new       -1.0435     1.4969 -0.6971   0.4866
```

```
print(round(summary(mdl_3)$r.squared, 4))
```

```
## [1] 0.9926
```

We see that the parameters are estimated close to the true values, even when X_1 and X_3 are related (though in this case - only approximately linearly). On the other hand, the coefficient of X_3 is not statistically significantly different form zero.

On the other hand, if we were to estimate the following model:

```
mdl_4 <- lm(y2_new ~ x1 + x3_new + I(x3_new^2))
print(round(coef(summary(mdl_4)), 4))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -41.4846    65.7724 -0.6307   0.5290
## x1           -8.1334    17.5430 -0.4636   0.6434
## x3_new      -11.0723    13.5328 -0.8182   0.4142
## I(x3_new^2)   0.1341     0.1798  0.7457   0.4568
```

Suddenly not only did **all** of our coefficients become insignificant, some of them even changed signs, while others significantly decreased.

Furthermore, from this model we cannot say **which** variables may be collinear, or even **if** there may be a collinearity problem.

What makes matters *worse* is the R^2 and the R^2_{adj} of this model is very large:

```
print(round(summary(mdl_4)$r.squared, 4))
```

```
## [1] 0.9926
```

```
print(round(summary(mdl_4)$adj.r.squared, 4))
```

```
## [1] 0.9925
```

We have that this model, with **all** insignificant coefficients, explains 99.3% of the variation in Y!

If we were to do the same on data without multicollinearity:

```
print(round(coef(summary(lm(y1 ~ x1 + x2))), 4))
```

```
##             Estimate Std. Error  t value Pr(>|t|)
## (Intercept)   2.2646     0.3334   6.7928        0
## x1            3.9838     0.0364 109.4794        0
## x2           -2.0507     0.0689 -29.7535        0
```

```
print(round(coef(summary(lm(y1 ~ x1 + x2 + I(x2^2)))), 4))
```

```
##             Estimate Std. Error  t value Pr(>|t|)
## (Intercept)   2.2197     0.3800   5.8406   0.0000
## x1            3.9838     0.0365 109.2182   0.0000
## x2           -1.9952     0.2341  -8.5237   0.0000
## I(x2^2)      -0.0132     0.0534  -0.2479   0.8045
```

Then we see that only the additional variable is insignificant. Furthermore, the additional variable did not significantly change the values of the remaining coefficients.

If the underlying model is **correctly** specified, multicollinearity still produces **unbiased** results. Only the **standard errors** are large in the related explanatory variables.

Testing for Multicollinearity

As we have mentioned, if $\det\left(\mathbf{X}^\top\mathbf{X}\right)$ is *close* to zero, then our data exhibits collinearity. Though there are a number of alternative ways to measure this presence, without examining only the design matrix:

- Large changes in the estimated regression coefficients when a predictor variable is added/removed are indicative of multicollinearity;
- If the multiple regression contains insignificant coefficients, then the F-test for significance should not reject the null hypothesis. If the null hypothesis for those coefficients is **rejected**, then we have an indication that the low t-values are due to multicollinearity.
- A popular measure of multicollinearity is the **variance inflation factor (VIF)**.

Variance Inflation Factor (VIF)

Variance Inflation Factor (VIF)
Consider the following multiple regression model:

$$Y_i = \beta_0 + \beta_1 X_{1,i} + ... + \beta_k X_{k,i} + \epsilon_k$$

Then, the variance inflation factor of $\widehat{\beta_j}$ is evaluated in the following steps:

1. Run the OLS **auxiliary regression** (i.e. a regression, whose coefficients are not of direct interest) of X_j as a function of all other explanatory variables:

$$X_{j,i} = \alpha_0 + \alpha_1 X_{1,i} + ... + \alpha_{j-1} X_{j-1,i} + \alpha_{j+1} X_{j+1,i} + ... + \alpha_k X_{k,i} + e_i$$

2. Then, the VIF of $\widehat{\beta_j}$ is:

$$\text{VIF}(\widehat{\beta_j}) = \frac{1}{1 - R_j^2}$$

where R_j^2 is the coefficient of determination of the regression in step (1).

3. Analyze the magnitude of multicollinearity from the size of $\text{VIF}(\widehat{\beta_j})$. as a **rule of thumb**:
   - if $\text{VIF}(\widehat{\beta_j}) > 10$, then multicollinearity is high.
   - if $\text{VIF}(\widehat{\beta_j}) < 5$, then the variable is not collinear with the remaining explanatory variables.

Finally: **the square root of the variance inflation factor indicates how much larger the standard error is, compared with what it would be if that variable were uncorrelated with the other predictor variables in the model**.

- As we have already seen, including the squared value of X_3 changed the sign and significance of **other** explanatory variables (which under assumptions **(MR.1)** - **(MR.5)** should be independent of X_3).
- Looking at the F-test in our model summary:

```
print(summary(mdl_4))
```

```
##
## Call:
## lm(formula = y2_new ~ x1 + x3_new + I(x3_new^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.78937 -0.58760  0.05885  0.68430  2.44656
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -41.4846    65.7724  -0.631    0.529
## x1           -8.1334    17.5430  -0.464    0.643
## x3_new      -11.0723    13.5328  -0.818    0.414
## I(x3_new^2)   0.1341     0.1798   0.746    0.457
##
## Residual standard error: 0.9693 on 196 degrees of freedom
## Multiple R-squared:  0.9926, Adjusted R-squared:  0.9925
## F-statistic:  8763 on 3 and 196 DF,  p-value: < 2.2e-16
```

We see that the p-value is **less** than the 0.05 significance level, meaning that we **reject** the null hypothesis that $\widehat{\beta}_1 = \widehat{\beta}_2 = \widehat{\beta}_3 = 0$ in the model $\widehat{Y}_i = \widehat{\beta}_0 + \widehat{\beta}_1 X_{1,i} + \widehat{\beta}_2 X_{3,i} + \widehat{\beta}_3 X_{3,i}^2$. On the other hand, the summary statistics show that we **do not reject** the **individual** hypothesis tests that $H_0 : \widehat{\beta}_j = 0$, $j = 1, 2, 3$.So, clearly something isn't right with the specified model.

- We can calculate the VIF manually:

```
# VIF(x3_new)
1/(1 - summary(lm(x3_new ~ 1 + x1))$r.squared)
```

```
## [1] 1563.274
```

```
# VIF(x1)
1/(1 - summary(lm(x1 ~ 1 + x3_new))$r.squared)
```

```
## [1] 1563.274
```

We can also use the built-in functions to verify that the variables are clearly collinear:

```
print(car::vif(lm(y2_new ~ x1 + x3_new)))
```

```
##       x1   x3_new
## 1563.274 1563.274
```

while for the model without collinearity the VIF is around 1 for each variable:

```
print(car::vif(mdl_1))
```

```
##       x1       x2
## 1.000768 1.000768
```

Note that this does not say anything about **causality** - whether X_1 influences X_3, or whether X_3 influences X_1 - we do not know.

Generalized Variance Inflation Factor (GVIF)

- Variance inflation factors are not fully applicable to models, which include a set of regressors (i.e. indicator regressors for the same categorical variable), or polynomial regressors.
- This is because the correlations among these variables are induced by the model structure and therefore are **artificial**. Usually, we are not concerned with artificial correlations of these types - we want to identify the effects of different explanatory variables.

Consequently, Fox, John, and Georges Monette. 1992. "Generalized Collinearity Diagnostics" introduced the **Generalized VIF**, denoted **GVIF**.

Assume that our regression model

$$\underset{N\times 1}{\mathbf{Y}} = \underset{N\times(k+1)}{\mathbf{X}} \underset{(k+1)\times 1}{\beta} + \underset{N\times 1}{\varepsilon}$$

can be written as:

$$\underset{N\times 1}{\mathbf{Y}} = \beta_0 + \underset{N\times r}{\mathbf{X}_1} \underset{r\times 1}{\beta_1} + \underset{N\times(k-r)}{\mathbf{X}_2} \underset{(k-r)\times 1}{\beta_2} + \underset{N\times 1}{\varepsilon}$$

where:

- $\mathbf{X}_1$ contains the **related** r indicator variables (or, a specific variable and its polynomial terms).
- $\mathbf{X}_2$ contains the **remaining** regressors, **excluding the constant**;

Then the **G**eneralized **VIF** is defined as:

$$\mathrm{GVIF}_1 = \frac{\det(\mathbf{R}_{11})\det(\mathbf{R}_{22})}{\det(\mathbf{R})}$$

where:

- $\mathbf{R}_{11}$ is the **correlation matrix** for $\mathbf{X}_1$;
- $\mathbf{R}_{22}$ is the **correlation matrix** for $\mathbf{X}_2$;
- $\mathbf{R}$ is the **correlation matrix** for all variables in the whole design matrix $\mathbf{X}$, **excluding the constant**;

The GVIF is calculated for sets of related regressors, such as a for a set of indicator regressors for some kind of categorical variable, or for polynomial variables:

- **For the continuous variables GVIF is the same as the VIF values before**;
- For the categorical variables, we now get one GVIF value for each separate category type (e.g. one value for all age groups, another value for all regional indicator variables and so on);

So, variables which *require* more than 1 coefficient and thus more than 1 degree of freedom are typically evaluated using the GVIF.

To make GVIFs comparable across dimensions, `Fox and Monette` also suggested using `GVIF^(1/(2*DF))`, where `DF` (degrees of freedom) is the number of coefficients in the subset. This reduces the GVIF to a linear measure. It is analogous to taking the square root of the usual VIF. In other words:

> We can apply the usual VIF rule of thumb if we *squared* the `GVIF^(1/(2*Df))` value. For example, requiring $\left(\text{GVIF}^{1/(2\cdot \text{DF})}\right)^2 < 5$ is equivalent to a requirement that $\text{VIF} < 5$ for the continuous (i.e. non-categorical) variables.

Example

Continuing our previous example, assume that we have the following age indicator variables, all of which are in a single variable 'age':

```
set.seed(123)
#
dt3$age <- sample(c("10_18", "18_26", "26_40", "other"), size = length(x1), replace = TRUE)
print(head(dt3))
```

```
##     y2_new        x1    x3_new   age
## 1 53.25364  6.879049 -11.90550 26_40
## 2 56.09227  7.539645 -12.55117 26_40
## 3 77.66881 11.117417 -15.91695 26_40
## 4 60.79827  8.141017 -13.13152 18_26
## 5 62.19339  8.258575 -13.24420 26_40
## 6 78.47473 11.430130 -16.20238 18_26
```

Now assume that we simply want to add it to our model with collinear variables:

```
mdl_5 <- lm(y2_new ~ x1 + x3_new + age, data = dt3)
print(round(coef(summary(mdl_5)), 4))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.7840     7.9300  0.7294   0.4666
## x1            4.6422     1.4438  3.2154   0.0015
## x3_new       -1.3184     1.5024 -0.8775   0.3813
## age18_26      0.1004     0.1943  0.5168   0.6059
## age26_40     -0.2383     0.1929 -1.2358   0.2180
## ageother     -0.0594     0.2080 -0.2854   0.7757
```

x1 and x3_new are continuous predictors and age is a categorical variable presented by the indicator/dummy variables age18_26, 26_40 and other. In this case 10_18 is the baseline, or base group.

```
print(cbind(DAAG::vif(mdl_5)))
```

```
##                [,1]
## x1       1579.3000
## x3_new   1579.7000
## age18_26    1.6115
## age26_40    1.6389
## ageother    1.5362
```

- The problem is that the VIF values are affected by the baseline of the categorical variable.
- In order to be sure of not having a VIF value above an acceptable level, it would be necessary to redo this analysis for every level of the categorical variable being the base group - i.e. we would also need to use `age18_26` as the base group and calculate the VIF's, then do the same with `age26_40` and finally with `other`. This gets more complicated if we have more than one categorical variable.

As such, it would be easier to calculate the GVIF's.

We begin by separating the indicator variables for the categorical age variable in a separate matrix $\mathbf{X}_1$ and the remaining explanatory variables (excluding the constant) in $\mathbf{X}_2$:

```
age <- dt3$age
X1 <- cbind(model.matrix(~ age + 0))
X2 <- cbind(dt3$x1, dt3$x3_new)
```

and here is how the matrices would look like:

```
print(head(X1))
```

```
##   age10_18 age18_26 age26_40 ageother
## 1        0        0        1        0
## 2        0        0        1        0
## 3        0        0        1        0
## 4        0        1        0        0
## 5        0        0        1        0
## 6        0        1        0        0
```

Note that since we do not include all of the indicator variables in the model so as to avoid the dummy variable trap, we need to drop one of the indicator variables. For simplicity, we will drop 10_18, since it was the base group for the previous model.

```
X1 <- X1[, -1]
print(head(X1))
```

```
##   age18_26 age26_40 ageother
## 1        0        1        0
## 2        0        1        0
## 3        0        1        0
## 4        1        0        0
## 5        0        1        0
## 6        1        0        0
```

```
print(head(X2, 4))
```

```
##           [,1]      [,2]
## [1,]  6.879049 -11.90550
## [2,]  7.539645 -12.55117
## [3,] 11.117417 -15.91695
## [4,]  8.141017 -13.13152
```

Now, we can calculate the GVIF for age:

```
tmp_gvif <- det(cor(X1)) * det(cor(X2)) / det(cor(cbind(X1, X2)))
tmp_gvif <- data.frame(GVIF = tmp_gvif)
tmp_gvif$"GVIF^(1/2Df)" <- tmp_gvif$GVIF^(1/(2 * (ncol(X1))))
print(tmp_gvif)
```

```
##       GVIF GVIF^(1/2Df)
## 1 1.015084     1.002498
```

We can also do the same for the continuous variables, say `x1`:

```
X1 <- cbind(x1)
X2 <- cbind(model.matrix(~ age + 0))[, -1]
X2 <- cbind(X2, x3_new)
#
tmp_gvif <- det(cor(X1)) * det(cor(X2)) / det(cor(cbind(X1, X2)))
tmp_gvif <- data.frame(GVIF = tmp_gvif)
tmp_gvif$"GVIF^(1/2Df)" <- tmp_gvif$GVIF^(1/(2 * (ncol(X1))))
print(tmp_gvif)
```

```
##      GVIF GVIF^(1/2Df)
## 1 1579.27     39.74003
```

Alternatively, we can do the same using the built-in functions:

```
print(car::vif(mdl_5))
```

```
##                GVIF Df GVIF^(1/(2*Df))
## x1     1579.270044  1       39.740031
## x3_new 1579.727037  1       39.745780
## age       1.015084  3        1.002498
```

and get identical calculation results.

Next, if we wanted to compare the values in the same sense as for the VIF, we need to square the `GVIF^(1/(2*Df))` values:

```
(car::vif(mdl_5)[, 3, drop = FALSE])^2
```

```
##        GVIF^(1/(2*Df))
## x1          1579.270044
## x3_new      1579.727037
## age            1.005003
```

We can then compare the values with the usual rule of thumb values of 5 and 10, as we did for VIF.

Note: we are still using the same `car::vif()` function - since we have included `age` - the `GVIF` is calculated automatically instead of the standard `VIF`.

Methods to Address Multicolinearity

The multicollinearity problem is that the collinear variables do not contain enough information about individual explanatory variable effects

There are a number of ways we can try to deal with multicollinearity:

- Make sure you have not fallen into the **dummy variable trap** - including a dummy variable for every category and including a constant term in the regression together guarantees perfect multicollinearity.
- **Do nothing** - leave the model as is, despite multicollinearity. If our purpose is to **predict** Y, the presence of multicollinearity doesn't affect the efficiency of extrapolating the fitted model to new data **as long as the predictor variables follow the same pattern of multicollinearity in the new data as in the data on which the regression model is based**. As mentioned before, if our sample is not representative, then we may get coefficients with incorrect signs, or our model may be overfitted.
- Try to **drop** some of the collinear variables. If our purpose is to **estimate the individual influence** of each explanatory variable, multicollinearity prevents us from doing so. However, **we lose additional information on Y from the dropped variables**. Furthermore, omission of relevant variables results in **biased** coefficient estimates of the remaining coefficients.
- Obtain more data. **This the ideal solution**. As we have seen from the OLS formulas, a larger sample size produces more precise parameter estimates (i.e. with smaller standard errors).

Methods to Address Multicolinearity (Cont.)

- Using **polynomial** models, or models with **interaction** terms may sometimes **lead to multicollinearity**, especially if those variables have a very limited attainable value range. A way to solve this would be to **mean-center** the predictor variables. On the other hand, if the variables do not have a limited range, then this transformation does not help.
- Alternatively, we may combine all (or some) multicollinear variables into groups and use the method of **principle components** (alternatively, **ridge regression**, **partial least squares regression** can also be used).
- If the variables with high VIF's are indicator variables, that represent a category with three or more categories, then high VIF may be caused when the base category has a small fraction of the overall observations. To address this, select the base category with a larger fraction of overall cases.
- Generally, if we are analyzing **time series data** - the correlated variables may in fact be the same variable, but at different lagged times. Consequently, including the relevant lags of the explanatory variables would account for such a problem. **For cross-sectional data however, this is not applicable**.

Example

We will show an example of how polynomial and interaction variables are strongly collinear, but centering the variables reduces this problem.
The simulated data is for the following model:

$$Y_i = \beta_0 + \beta_1 X_{1,i} + \beta_2 X_{1,i}^2 + \beta_3 X_{2,i} + \beta_4 \left(X_{1,i} \times X_{2,i}\right) + \beta_5 X_{3,i} + \epsilon_i$$

```r
set.seed(123)
#
N <- 1000
beta_vec <- c(5, 2, 0.002, 3, -0.05, 4)
#
x1 <- rnorm(mean = 15, sd = 5, n = N)
x2 <- sample(1:20, size = N, replace = TRUE)
x3 <- sample(seq(from = 0, to = 12, length.out = 50), size = N, replace = TRUE)
e  <- rnorm(mean = 0, sd = 3, n = N)
#
x_mat <- cbind(1, x1, x1^2, x2, x1 * x2, x3)
#
y <- x_mat %*% beta_vec + e
#
dt_sq <- data.frame(y, x1, x2, x3)
```

Our estimated model is:

```
mdl_sq <- lm(y ~ x1 + I(x1^2) + x2 + x1:x2 + x3, data = dt_sq)
print(round(coef(summary(mdl_sq)), 5))
```

```
##             Estimate Std. Error   t value Pr(>|t|)
## (Intercept)  4.84627    0.87003   5.57024  0.00000
## x1           2.08893    0.09254  22.57284  0.00000
## I(x1^2)     -0.00147    0.00267  -0.55039  0.58217
## x2           2.93555    0.05116  57.37500  0.00000
## x3           3.96010    0.02542 155.80122  0.00000
## x1:x2       -0.04594    0.00328 -13.98404  0.00000
```

We see that the squared term may be insignificant. If we examine the VIF:

```
print(car::vif(mdl_sq))
```

```
##        x1   I(x1^2)        x2        x3     x1:x2
## 25.641296 20.849557  9.806743  1.003584 12.685321
```

We see that the VIF is very large, despite the fact that only one variable appears to be insignificant. Furthermore, the signs of the coefficients are close to the true ones.

One of the strategies of dealing with multicollinearity in polynomial and interaction terms is to initially **center** the values by subtracting their means and then create polynomial and interaction terms. The mean of X_1 and X_2 is:

```
print(colMeans(dt_sq))
```

```
##         y        x1        x2        x3
## 83.299219 15.080639 10.493000  5.990694
```

One efficient way of subtracting the means is by applying a function to the data:

```
dt_sq_centered <- dt_sq
print(head(dt_sq_centered, 5))
```

```
##           y       x1 x2         x3
## 1  80.27734 12.19762  1 12.0000000
## 2 111.49284 13.84911 17  9.3061224
## 3 108.27019 22.79354 18  6.6122449
## 4  58.86745 15.35254 10  0.9795918
## 5 102.96509 15.64644  9 11.2653061
```

```
dt_sq_centered[, c("x1", "x2")] <- apply(dt_sq_centered[, c("x1", "x2")], MARGIN = 2, function(x){x -
print(head(dt_sq_centered, 5))
```

```
##           y          x1     x2         x3
## 1  80.27734 -2.8830176 -9.493 12.0000000
## 2 111.49284 -1.2315268  6.507  9.3061224
## 3 108.27019  7.7129022  7.507  6.6122449
## 4  58.86745  0.2719026 -0.493  0.9795918
## 5 102.96509  0.5657993 -1.493 11.2653061
```

Finally, if we re-estimate the model with the centered variables:

```
mdl_sq_centered <- lm(formula(mdl_sq), data = dt_sq_centered)
print(round(coef(summary(mdl_sq_centered)), 5))
```

```
##             Estimate Std. Error   t value Pr(>|t|)
## (Intercept) 59.54801    0.18804 316.68340  0.00000
## x1           1.56258    0.01833  85.23459  0.00000
## I(x1^2)     -0.00147    0.00267  -0.55039  0.58217
## x2           2.24278    0.01642 136.62322  0.00000
## x3           3.96010    0.02542 155.80122  0.00000
## x1:x2       -0.04594    0.00328 -13.98404  0.00000
```

Note: the standard errors, t-statistics and p-values remain the same. However, if we were to check the collinearity:

```
print(car::vif(mdl_sq_centered))
```

```
##       x1  I(x1^2)       x2       x3    x1:x2
## 1.006275 1.009893 1.009524 1.003584 1.012698
```

We see that it has disappeared.

> So, in case of polynomial and interaction variables, their multicollinearity has no adverse consequences.

On the other hand, if there are many other variables, then it may be difficult to distinguish, which variables are affected by multicollinearity.

Consequently, it is better to determine the collinearity between different variables **first**, and then add polynomial and interaction terms **later**.

Furthermore, it is worth noting, that when we center the independent variable, the interpretation of its coefficient remains the same.

On the other hand, the intercept parameter changes - β_0 can now be interpreted as the **average** value of Y, when X is equal to the average in the sample (when X is equal to the sample mean - its centered variable is zero, therefore the associated β_j also becomes zero).

Finally:

From a set of multiple linear regression models, defined as:

$$Y_i = \beta_0 + \beta_1 X_{1,i} + ... + \beta_k X_{k,i} + \epsilon_i$$

The best regression models are those where the predictor variables X_j:

- each highly correlate with the dependent variable Y;
- minimally correlate with each other (preferably - no correlation);

Weak collinearity is not a violation of OLS assumptions. If the estimated coefficients:

- have the expected signs and magnitudes;
- they are not sensitive to adding or removing a few observations
- they are not sensitive to adding or removing insignificant variables;

then there is no reason to try to identify and mitigate collinearity.