PE I: Multivariable Regression

Model Specification, OLS, Confidence Intervals, Hypothesis Testing & GoF
(Chapters 4.2 & 4.3)

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

Multiple Regression

We can augment our initial univariate regression, by including:

- additional explanatory variables;
- including polynomial (squared, cubed, etc.) variables;
- including interaction terms;
- including dummy variables.

For example, we could specify our model as follows:

$$\log(Y_i) = \beta_0 + \beta_1 X_{1,i} + \beta_2 \log(X_{2,i}) + \beta_3 D_{1,i} + \beta_4 X_{1,i}^2 + \beta_5 [\log(X_{2,i}) \times D_{1,i}] + \epsilon_i, \quad i = 1, ..., N$$

where $D_{1,j}$ is some kind of dummy variable.

See the lecture notes and previous slides on interpretation of polynomial, interaction and dummy variables.

Then, we can re-write the model in matrix notation:

$$\begin{bmatrix} \log(Y_1) \\ \log(Y_2) \\ \vdots \\ \log(Y_N) \end{bmatrix} = \begin{bmatrix} 1 & X_{1,1} & \log(X_{2,1}) & D_{1,1} & X_{1,1}^2 & \log(X_{2,1}) \times D_{1,1} \\ 1 & X_{1,2} & \log(X_{2,2}) & D_{1,2} & X_{1,2}^2 & \log(X_{2,2}) \times D_{1,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & X_{1N} & \log(X_{2,N}) & D_{1,N} & X_{1,N}^2 & \log(X_{2,N}) \times D_{1,N} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_N \end{bmatrix}$$

or in a more compact form:

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon$$

Note that sometimes the explanatory variable matrix $\mathbf{X}$ is called the **design matrix**, or the model matrix, or regressor matrix.

Dummy Variables from Categorical Variables

Consider a dataset, where the observations $(Y_j, X_{1,j}, ..., X_{k,j})$, $j = 1, ..., N$ are collected at specific days of the week. This in turn gives us a variable, which specifies the days of the week:

$$day\_of\_the\_week_j = \begin{cases} \text{Monday} & , \text{ if } j = \text{Monday} \\ \text{Tuesday} & , \text{ if } j = \text{Tuesday} \\ & \vdots \\ \text{Sunday} & , \text{ if } j = \text{Sunday} \end{cases}$$

```
set.seed(123)
N <- 100
day_vec_ordered <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")
data <- data.frame(day_of_the_week = sample(day_vec_ordered, size = N, replace = TRUE))
print(t(head(data, 5)))
```

```
##                 1        2        3           4          5
## day_of_the_week "Sunday" "Sunday" "Wednesday" "Saturday" "Wednesday"
```

```
print(head(data$day_of_the_week, 5))
```

```
## [1] Sunday    Sunday    Wednesday Saturday  Wednesday
## Levels: Friday Monday Saturday Sunday Thursday Tuesday Wednesday
```

We may want to change the **ordering** of the factors:

```
data$day_of_the_week <- factor(data$day_of_the_week, levels = day_vec_ordered)
print(head(data$day_of_the_week))
```

```
## [1] Sunday    Sunday    Wednesday Saturday  Wednesday Tuesday
## Levels: Monday Tuesday Wednesday Thursday Friday Saturday Sunday
```

We can create the design matrix **with a constant term column** and **without the base group** in the following way:

```
x_mat <- model.matrix(~ day_of_the_week, data)
print(head(x_mat))
```

```
##   (Intercept) day_of_the_weekTuesday day_of_the_weekWednesday
## 1           1                      0                        0
## 2           1                      0                        0
## 3           1                      0                        1
## 4           1                      0                        0
## 5           1                      0                        1
## 6           1                      1                        0
##   day_of_the_weekThursday day_of_the_weekFriday day_of_the_weekSaturday
## 1                       0                     0                       0
## 2                       0                     0                       0
## 3                       0                     0                       0
## 4                       0                     0                       1
## 5                       0                     0                       0
## 6                       0                     0                       0
##   day_of_the_weekSunday
## 1                     1
## 2                     1
## 3                     0
```

```
print(head(cbind(data, x_mat[, -1])))
```

```
##   day_of_the_week day_of_the_weekTuesday day_of_the_weekWednesday
## 1          Sunday                      0                        0
## 2          Sunday                      0                        0
## 3       Wednesday                      0                        1
## 4        Saturday                      0                        0
## 5       Wednesday                      0                        1
## 6         Tuesday                      1                        0
##   day_of_the_weekThursday day_of_the_weekFriday day_of_the_weekSaturday
## 1                       0                     0                       0
## 2                       0                     0                       0
## 3                       0                     0                       0
## 4                       0                     0                       1
## 5                       0                     0                       0
## 6                       0                     0                       0
##   day_of_the_weekSunday
## 1                     1
## 2                     1
## 3                     0
## 4                     0
## 5                     0
## 6                     0
```

If needed, we can create a matrix of **only our dummy variables**, but we need to be mindful of not including them all in a model with a constant term (otherwise we will fall in the dummy variable trap):

```
tmp_mat <- model.matrix(~ -1 + day_of_the_week, data)
print(head(tmp_mat, 5))
```

```
##   day_of_the_weekMonday day_of_the_weekTuesday day_of_the_weekWednesday
## 1                     0                      0                        0
## 2                     0                      0                        0
## 3                     0                      0                        1
## 4                     0                      0                        0
## 5                     0                      0                        1
##   day_of_the_weekThursday day_of_the_weekFriday day_of_the_weekSaturday
## 1                       0                     0                       0
## 2                       0                     0                       0
## 3                       0                     0                       0
## 4                       0                     0                       1
## 5                       0                     0                       0
##   day_of_the_weekSunday
## 1                     1
## 2                     1
## 3                     0
## 4                     0
## 5                     0
```

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:

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon \qquad \text{(MR.1)}$$

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

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

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:

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

OLS Estimation, Confidence Intervals and Hypothesis Testing

Most of the properties and formula expressions presented in this chapter are identical to the simple univariate regression case.

The multiple regression model in this section is specified in the following matrix form:

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon$$

We will further assume that the multiple regression assumptions (MR.1) - (MR.6) hold true.

Example

We will use the following model to aid our presented methodology:

$$\log(Y_i) = \beta_0 + \beta_1 X_{1i} + \beta_2 \log(X_{2i}) + \beta_3 \text{MARRIED}_i + \beta_4 \text{AGE\_GROUP}_{1i} + \beta_5 \text{AGE\_GROUP}_{2i} + \beta_6(\text{AGE\_GROUP}_{2i} \times X_{1i}) + \beta_7(\text{MARRIED}_i \times \text{AGE\_GROUP}_{1i}) + \epsilon_i$$

where $MARRIED_i = 1$, if the i-th person is married, 0 otherwise; AGE_GROUP_{ji} are different age groups: if $j = 1$ - between $20 - 30$; if $j = 2$ - between $31 - 65$, the **base group**, AGE_GROUP_{OTHER}, consists the people with ages in the remaining age brackets, not covered by $j = 1, 2$.

In this example the specified model has several distinctions:

- The dependent variable Y is log-transformed;
- Some independent variables are log-transformed;
- Inclusion of indicator variables;
- Cross-products (interaction terms) of some independent variables;
- Not all indicator variable cross-products have a significant effect - for example $(\text{MARRIED}_i \times \text{AGE\_GROUP}_{2i})$ is not included, which means that married people, aged $31 - 65$ do not have any additional effects on $\log(Y_i)$, compared to non-married people in the base age group.

See the lecture notes on an example on how to simulate such a model.

Assume that we have either created the age group dummy variables using the previously mentioned methods, or that these variables were already provided in the dataset:

```
head(data_mat)
```

```
##           y        x1       x2 married age_gr1 age_gr2  age_group
## 1 29.313052 10.206100 2.450563       1       0       1 aged_31_65
## 2 14.656237 10.164646 2.976220       0       0       1 aged_31_65
## 3 28.228720 10.816951 2.255319       1       0       0      other
## 4  7.741518 11.713026 4.286608       1       0       1 aged_31_65
## 5  9.333522  6.220738 2.514393       1       1       0 aged_20_30
## 6  2.165643  5.813722 4.237797       1       0       1 aged_31_65
```

```
print(head(data_mat$age_group))
```

```
## [1] aged_31_65 aged_31_65 other      aged_31_65 aged_20_30 aged_31_65
## Levels: aged_20_30 aged_31_65 other
```

We may also want to re-level the **categorical (factor) variable** so that the age group - `other` - would be the base level (this is equivalent to being the first level):

```
data_mat$age_group <- relevel(data_mat$age_group, "other")
print(head(data_mat$age_group))
```

```
## [1] aged_31_65 aged_31_65 other      aged_31_65 aged_20_30 aged_31_65
## Levels: other aged_20_30 aged_31_65
```

OLS Estimation of Regression Parameters

We want to minimize the sum of squared residuals:

$$\begin{aligned} RSS(\boldsymbol{\beta}) &= \boldsymbol{\varepsilon}^\top \boldsymbol{\varepsilon} \\ &= (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^\top (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) \\ &= \mathbf{Y}^\top \mathbf{Y} - \boldsymbol{\beta}^\top \mathbf{X}^\top \mathbf{Y} - \mathbf{Y}^\top \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\beta}^\top \mathbf{X}^\top \mathbf{X}\boldsymbol{\beta} \rightarrow \min_{\beta_0, \beta_1, \ldots, \beta_k} \end{aligned}$$

Then, equating the partial derivative to zero:

$$\frac{\partial RSS(\widehat{\boldsymbol{\beta}})}{\partial \widehat{\boldsymbol{\beta}}} = -2\mathbf{X}^\top \mathbf{Y} + 2\mathbf{X}^\top \mathbf{X}\widehat{\boldsymbol{\beta}} = 0$$

gives us the **OLS estimator**:

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y} \qquad \text{(OLS)}$$

The **Gauss-Markov Theorem** for the multiple regression states that: if the conditions (MR.1) - (MR.5) hold true, the OLS estimators $\widehat{\boldsymbol{\beta}}$ are the **B**est **L**inear **U**nbiased **E**stimators and they are consistent (BLUE&C) with the true parameter values of the multiple regression model.

The **proofs** are analogous to the proofs in the simple univariate regression case since we are using the same matrix notation as before!

This means that the variance-covariance matrix of the OLS estimator vector is:

$$\mathbb{V}\text{ar}(\widehat{\boldsymbol{\beta}}) = \begin{bmatrix} \mathbb{V}\text{ar}(\widehat{\beta}_0) & \mathbb{C}\text{ov}(\widehat{\beta}_0, \widehat{\beta}_1) & \dots & \mathbb{C}\text{ov}(\widehat{\beta}_0, \widehat{\beta}_k) \\ \mathbb{C}\text{ov}(\widehat{\beta}_1, \widehat{\beta}_0) & \mathbb{V}\text{ar}(\widehat{\beta}_1) & \dots & \mathbb{C}\text{ov}(\widehat{\beta}_1, \widehat{\beta}_k) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{C}\text{ov}(\widehat{\beta}_k, \widehat{\beta}_0) & \mathbb{C}\text{ov}(\widehat{\beta}_k, \widehat{\beta}_1) & \dots & \mathbb{V}\text{ar}(\widehat{\beta}_k) \end{bmatrix} = \sigma^2 \left(\mathbf{X}^\top \mathbf{X}\right)^{-1}$$

The difference from the univariate case is in the estimation of the unknown error variance parameter σ^2.

```r
x_mat_1 <- cbind(1, data_mat$x1, log(data_mat$x2), data_mat$married,
                 data_mat$age_gr1, data_mat$age_gr2,
                 data_mat$age_gr2 * x1, data_mat$age_gr1 * data_mat$married)
colnames(x_mat_1) <- c("intercept", "x1", "log_x2", "married",
                       "age_gr1", "age_gr2", "age_gr2_x1", "married_age_gr1")
head(x_mat_1)
```

```
##      intercept        x1    log_x2 married age_gr1 age_gr2 age_gr2_x1
## [1,]         1 10.206100 0.8963179       1       0       1  10.206100
## [2,]         1 10.164646 1.0906541       0       0       1  10.164646
## [3,]         1 10.816951 0.8132915       1       0       0   0.000000
## [4,]         1 11.713026 1.4554958       1       0       1  11.713026
## [5,]         1  6.220738 0.9220314       1       1       0   0.000000
## [6,]         1  5.813722 1.4440436       1       0       1   5.813722
##      married_age_gr1
## [1,]               0
## [2,]               0
## [3,]               0
## [4,]               0
## [5,]               1
## [6,]               0
```

```r
beta_est <- solve(t(x_mat_1) %*% x_mat_1) %*% t(x_mat_1) %*% log(data_mat$y)
colnames(beta_est) <- "coef_est"
print(beta_est)
```

```
##                    coef_est
## intercept         4.01017653
## x1                0.15906399
## log_x2           -3.00236892
## married           0.04664335
## age_gr1           0.02473384
## age_gr2          -0.14682506
## age_gr2_x1        0.05036566
## married_age_gr1 -0.02678704
```

In comparison, the true, **unknown** parameters were:

```
## [1]  4.00  0.16 -3.00  0.05  0.02 -0.15  0.05 -0.03
```

Estimation of The Error Variance Parameter

Define the residual vector as the difference between the actual and the fitted values:

$$\widehat{\varepsilon} = [\widehat{\epsilon}_1, ..., \widehat{\epsilon}_N]^\top = \mathbf{Y} - \widehat{\mathbf{Y}} = \mathbf{Y} - \mathbf{X}\widehat{\beta}$$

Then:

An estimator of σ^2, that uses the information from $\widehat{\epsilon}_i^2$ is:

$$\widehat{\sigma}^2 = \frac{1}{N-(k+1)} \sum_{i=1}^{N} \widehat{\epsilon}_i^2 = \frac{\widehat{\varepsilon}^\top \widehat{\varepsilon}}{N-(k+1)}$$

where $k+1$ is the number of parameters in $\widehat{\beta} = \left[\widehat{\beta}_0, \widehat{\beta}_1, ..., \widehat{\beta}_k\right]^\top$.

Having estimated the unknown parameters via OLS and the variance parameters allows us to calculate various confidence intervals, as discussed in the univariate regression case.

If, additionally to (MR.1) - (MR.5), condition (MR.6) holds true, the conditional distribution of the OLS estimators is:

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

This allows us to calculate the confidence intervals for our parameters, their predicted values and the mean response.

Important!

It should be noted that we require $N > k + 1$. That is, as long as we have more data points than unknown parameters, we will be able to carry out OLS estimation and calculate any relevant test statistics or confidence intervals.

- If $N = k + 1$, then our standard errors are very large. This is because standard errors reflect the degree of uncertainty of our estimates and there is less information per parameter to get an accurate estimate (remember that $\widehat{Y}$ is the **mean** response).
- If N is close to $k + 1$, then we would need to reduce the number of parameters in our model.

Another way to look at this is by examining the system of equations of the multivariable regression - the case of $N < k + 1$ means that we have more unknown parameters, $k + 1$, than we have equations, N - hence our system of equations has infinitely many solutions (or, in a rare case - no solution).

Ideally, as a rule of thumb, we would need at least 20 - 30 observations for every parameter that we want to estimate. On the other hand, while the sample size itself is important, a more pressing concern is whether the sample is representative of the overall population.

For our example dataset, the variance can be easily estimated from the residuals:

```
y_fit <- x_mat_1 %*% beta_est
#
resid <- log(data_mat$y) - y_fit
#
sigma2_est <- sum(resid^2) / (nrow(data_mat) - length(beta_est))
print(paste0("Estimated residual standard error: ", round(sqrt(sigma2_est), 5)))
```

```
## [1] "Estimated residual standard error: 0.04951"
```

Parameter Confidence Intervals

The $100 \cdot (1-\alpha)\%$ interval estimate for parameter β_i, $i = 0, ..., k$ is calculated in the same way as for the simple univariate regression:

$$\left[\widehat{\beta}_i - t_c \cdot \text{se}(\widehat{\beta}_i),\ \widehat{\beta}_i + t_c \cdot \text{se}(\widehat{\beta}_i)\right]$$

where $t_c = t_{(1-\alpha/2, N-(k+1))}$ and $\text{se}(\widehat{\beta}_i) = \sqrt{\widehat{\mathbb{V}\text{ar}}(\widehat{\beta}_\mathbf{i})}$. If this interval estimator is used in many samples from the population, then $100 \cdot (1-\alpha)\%$ of them will contain the true parameter β_i.

Alternatively, we might introduce some kind of non-sample information on the coefficients in the form of **linear restrictions**.

Interval Estimation for a Linear Combination of Coefficients

In general, a linear combination of coefficients can be specified as:

$$\sum_{j=0}^{k} c_j \widehat{\beta}_j = c_0 \widehat{\beta}_0 + ... + c_k \widehat{\beta}_k = \widehat{r}$$

and has the following variance:

$$\widehat{\mathbb{V}\mathrm{ar}}\left(\widehat{r}\right) = \widehat{\mathbb{V}\mathrm{ar}}\left(\sum_{j=0}^{k} c_j \widehat{\beta}_j\right) = \sum_{j=0}^{k} c_j^2 \cdot \widehat{\mathbb{V}\mathrm{ar}}\left(\widehat{\beta}_j\right) + 2 \cdot \sum_{i<j} c_i c_j \cdot \widehat{\mathbb{C}\mathrm{ov}}\left(\widehat{\beta}_i,\ \widehat{\beta}_j\right)$$

This allows estimating the confidence interval of the specified linear combination as:

$$\left[\widehat{r} - t_c \cdot \mathrm{se}(\widehat{r}),\ \widehat{r} + t_c \cdot \mathrm{se}(\widehat{r})\right]$$

where $t_c = t_{(1-\alpha/2, N-(k+1))}$

We are interested in different parameter linear combinations when we are considering the mean response $\mathbb{E}(Y|\mathbf{X} = \mathbf{X}_0)$ for some explanatory variables, $\mathbf{X}_0$. Alternatively, we may be interested in the effect of changing **two or more** explanatory variables **simultaneously**.
Finally, parameter linear combinations are especially relevant if the effect of an explanatory variable depends on two or more parameters - i.e. in models with polynomial variables, or models with interaction terms.

Mean Response Confidence Intervals

The mean response estimator $\widehat{\mathbb{E}}(\mathbf{Y}|\mathbf{X} = \mathbf{X}_0) = \widehat{\mathbf{Y}} = \mathbf{X}_0\widehat{\boldsymbol{\beta}}$ follows the same distribution as highlighted in the simple univariate regression:

$$\left(\widehat{\mathbf{Y}}|\mathbf{X}_0, \mathbf{X}\right) \sim \mathcal{N}\left(\mathbf{X}_0\boldsymbol{\beta}, \quad \sigma^2\mathbf{X}_0\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}_0^\top\right)$$

which means that the $100 \cdot (1-\alpha)\%$ confidence intervals for the mean response are:

$$\left[\widehat{Y}_i - t_{(1-\alpha/2, N-(k+1))} \cdot \text{se}(\widehat{Y}_i), \;\; \widehat{Y}_i + t_{(1-\alpha/2, N-(k+1))} \cdot \text{se}(\widehat{Y}_i)\right]$$

where $\text{se}(\widehat{Y}_i) = \sqrt{\widehat{\mathbb{V}\text{ar}}(\widehat{Y}_i)}$ is the square root of the corresponding i-th diagonal element of $\widehat{\mathbb{V}\text{ar}}(\widehat{\mathbf{Y}}) = \widehat{\sigma}^2\mathbf{X}_0\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}_0^\top$, where $\widehat{\sigma}^2$ is estimated for the multiple regression model case as

Prediction Intervals

Following the simple univariate regression case, the $100 \cdot (1-\alpha)\%$ **prediction interval** for $\widehat{Y}_i$ is:

$$\left[\widehat{Y}_i - t_{(1-\alpha/2, N-(k+1))} \cdot \text{se}(\widetilde{e}_i),\ \ \widehat{Y}_i + t_{(1-\alpha/2, N-(k+1))} \cdot \text{se}(\widetilde{e}_i)\right]$$

where the *standard forecast error* $\text{se}(\widetilde{e}_i) = \sqrt{\widehat{\mathbb{V}\text{ar}}(\widetilde{e}_i)}$ is the square root of the corresponding i-th diagonal element of $\widehat{\mathbb{V}\text{ar}}(\widetilde{\boldsymbol{e}}) = \widehat{\sigma}^2 \left(\mathbf{I} + \mathbf{X}_0 \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}_0^\top\right)$

Hypothesis Testing

Testing For Significance of a Single Parameter

If we wanted to test a hypothesis for β_j:

$$\begin{cases} H_0 & : \widehat{\beta}_j = c \\ H_1 & : \widehat{\beta}_j \neq c \quad (\text{or } < c, \quad \text{or } > c) \end{cases}$$

We firstly need to calculate the t-statistic:

$$t_j = \frac{\widehat{\beta}_j - c}{\text{s.e}(\widehat{\beta}_j)} \sim t_{(N-(k+1))}$$

where $\text{s.e}(\widehat{\beta}_j) = \sqrt{\widehat{\mathbb{V}\text{ar}}(\widehat{\beta}_0)}$ and $\widehat{\mathbb{V}\text{ar}}(\widehat{\beta}_j)$ is the corresponding diagonal element from $\widehat{\mathbb{V}\text{ar}}(\widehat{\boldsymbol{\beta}}) = \widehat{\sigma}^2 \left(\mathbf{X}^\top \mathbf{X}\right)^{-1}$.

While the formula for the t-statistic remains the same, but its distribution depends on the number of estimated unknown parameters - a t distribution with $N - (k + 1)$ degrees of freedom.

The critical value t_c also depends on the number of estimated parameters:

- If the alternative is $H_1 : \widehat{\beta}_j > c$, we reject H_0 and accept the alternative H_1, if $t_i \geq t_c$, where $t_c = t_{(1-\alpha, N-(k+1))}$;
- If the alternative is $H_1 : \widehat{\beta}_j < c$, we reject H_0 and accept the alternative H_1, if $t_i \leq t_c$, where $t_c = t_{(\alpha, N-(k+1))}$;
- If the alternative is $H_1 : \widehat{\beta}_j \neq c$, we reject H_0 and accept the alternative H_1, if $t_i \leq t_{(\alpha/2, N-(k+1))}$, or $t_i \geq t_{(1-\alpha/2, N-(k+1))}$;
- We can also calculate the associated p-value: if $p \leq \alpha$, we reject H_0; if $p \geq \alpha$, we do not reject H_0;

Example

Continuing our example, we want to test the two-tail hypothesis $H_0 : \beta_j = 0$ against the alternative $H_1 : \beta_j \neq 0$ for each coefficient $j = 0, ..., 7$ for $\alpha = 0.05$ significance level. Note that we have always used the matrix notation - this is helpful when we want to calculate the results for multiple values:

```r
#Parameter variance-covariance matrix
beta_vcov <- sigma2_est * solve(t(x_mat_1) %*% x_mat_1)
# Calculate the test statistic for each coefficient:
t_stat <- beta_est / sqrt(diag(beta_vcov))
# Calculate the associated p-value:
p_val <- 2 * pt(-abs(t_stat), df = nrow(data_mat) - length(beta_est), lower = TRUE)
# Combine the output:
test_res <- cbind(t_stat, p_val)
colnames(test_res) <- c("t_stat", "p_val")
print(test_res)
```

```
##                      t_stat         p_val
## intercept        326.510512  0.000000e+00
## x1               169.415700  0.000000e+00
## log_x2          -502.842822  0.000000e+00
## married           12.035132  3.091084e-31
## age_gr1            4.776168  2.055881e-06
## age_gr2           -8.620667  2.598470e-17
## age_gr2_x1        30.619696 1.744506e-145
## married_age_gr1   -4.054822  5.410364e-05
```

We see that all of the specified coefficients are statistically significantly different from zero since we do not reject the null hypothesis for each of the coefficients.

Example

Additionally, we want to test the following hypothesis:

A one percent increase in X_2 would result in a 3% reduction in Y

which can be written as the following hypothesis:

$$\begin{cases} H_0 & : \beta_2 = -3 \\ H_1 & : \beta_2 \neq -3 \end{cases}$$

```
# Calculate the test statistic for beta[2] each coefficient:
t_stat <- (beta_est["log_x2", ] - (-3)) / sqrt(diag(beta_vcov)["log_x2"])
# Calculate the associated p-value:
p_val <- 2 * pt(-abs(t_stat), df = nrow(data_mat) - length(beta_est), lower = TRUE)
# combine the output
hyp_b2 <- cbind(t_stat, p_val)
print(hyp_b2)
```

```
##            t_stat     p_val
## log_x2 -0.3967519 0.6916357
```

Since the $p - value > 0.05$, we have no grounds to reject the null hypothesis and conclude that β_2 is not statistically significantly different from -3.

Joint Hypothesis Test for Multiple Coefficient Significance

The hypothesis tests with t-statistics allow for testing of a single equality in the null hypothesis. But what if we want to test a **joint** hypothesis, where multiple values were to be equal to some values?

One of the more popular types of joint hypothesis tests involves checking whether a group of variables is statistically significantly different from zero in a particular model.

If we wanted to test, whether M coefficients with index $i_1, ..., i_M \in \{0, 1, ..., k\}$ are statistically significantly different from zero, we would specify the following hypothesis:

$$\begin{cases} H_0 & : \beta_{i_1} = 0, \ \beta_{i_2} = 0, \ ..., \ \beta_{i_M} = 0 \\ H_1 & : \beta_{i_j} \neq 0, \quad \text{for some } j \end{cases}$$

We can test the hypothesis with an F-**test**, which evaluates, whether a reduction in the residual sum of squares (**RSS**) is significantly different.

If adding the additional variables does not significantly reduce the residual sum of squares, then those variable contribute little to the explanation of the variation in the dependent variable and we would not reject the null hypothesis.

Define the following **RSS**:

- **RSS**$_{UR}$ - the residual sum of squares of the **unrestricted** (i.e. the **full**) model under the alternative hypothesis. The coefficient of determination in the unrestricted model is R^2_{UR};
- **RSS**$_R$ - the residual sum of squares of the **restricted** model under the null hypothesis (i.e. when some of the parameters are not statistically significantly different from zero). The coefficient of determination in the restricted model is R^2_R;

Then, the F-statistic is given by:

$$F = \frac{(\text{RSS}_R - \text{RSS}_{UR})/M}{\text{RSS}_{UR}/(N-(k+1))} = \frac{(R^2_{UR} - R^2_R)/M}{(1-R^2_{UR})/(N-(k+1))} \sim F_{(M,N-(k+1))}$$

We then select the significance level α and calculate the critical value $F_c = F_{(1-\alpha,M,N-(k+1))}$.

If $F \geq F_c$, we **reject the null hypothesis** and conclude that **at least one** of the coefficients in the null is not zero. We can also calculate the associated p-value.

Example

We again turn to our example data, where we will estimate an **unrestricted** model with an additional coefficients. For interest, we also calculate the t-statistic and p-values for tests of individual coefficient significance:

$$\begin{aligned}\log(Y_i) = \beta_0 &+ \beta_1 X_{1i} + \beta_2 \log(X_{2i}) + \beta_3 \text{MARRIED}_i + \beta_4 \text{AGE\_GROUP}_{1i} + \beta_5 \text{AGE\_GROUP}_{2i} \\ &+ \beta_6(\text{AGE\_GROUP}_{2i} \times X_{1i}) + \beta_7(\text{MARRIED}_i \times \text{AGE\_GROUP}_{1i}) \\ &+ \beta_8(\text{MARRIED}_i \times \text{AGE\_GROUP}_{2i}) + \beta_9(\text{AGE\_GROUP}_{1i} \times X_{1i}) + \epsilon_i\end{aligned}$$

We begin by creating the appropriate **design matrix**:

```
x_mat_UR <- cbind(1, data_mat$x1, log(data_mat$x2), data_mat$married,
                  data_mat$age_gr1, data_mat$age_gr2,
                  data_mat$age_gr2 * x1, data_mat$age_gr1 * data_mat$married,
                  data_mat$age_gr2 * data_mat$married, data_mat$age_gr1 * x1)
colnames(x_mat_UR) <- c(colnames(x_mat_1), "married_age_gr2", "age_gr1_x1")
```

we then estimate the model coefficients and their variance-covariance matrix:

```
# parameter estimation
beta_est_UR <- solve(t(x_mat_UR) %*% x_mat_UR) %*% t(x_mat_UR) %*% log(data_mat$y)
colnames(beta_est_UR) <- "coef_est"
# fitted value and error calculation
y_fit_UR <- x_mat_UR %*% beta_est_UR
resid_UR <- log(data_mat$y) - y_fit_UR
sigma2_est_UR <- sum(resid_UR^2) / (nrow(data_mat) - length(beta_est_UR))
beta_vcov_UR <- sigma2_est_UR * solve(t(x_mat_UR) %*% x_mat_UR)
```

and finally perform significance testing separately for each estimated parameter:

```
# significance testing of individual parameters
t_stat_UR <- beta_est_UR / sqrt(diag(beta_vcov_UR))
p_val_UR <- 2 * pt(-abs(t_stat_UR), df = nrow(data_mat) - length(beta_est_UR), lower = TRUE)
# combined output
mdl_UR_out <- cbind(beta_est_UR, t_stat_UR, p_val_UR)
#
print(round(mdl_UR_out, 5))
```

```
##                 coef_est   coef_est coef_est
## intercept        4.02536  258.76200  0.00000
## x1               0.15772  119.40341  0.00000
## log_x2          -3.00231 -502.99025  0.00000
## married          0.04293    7.83446  0.00000
## age_gr1         -0.00368   -0.18757  0.85125
## age_gr2         -0.16403   -8.21466  0.00000
## age_gr2_x1       0.05174   27.37776  0.00000
## married_age_gr1 -0.02278   -2.97211  0.00303
## married_age_gr2  0.00702    0.90482  0.36578
## age_gr1_x1       0.00266    1.41740  0.15668
```

Again, we will consider the above, `mdl_UR_out`, as the **unrestricted** model.

The **restricted** model will is the true model regression which we have previously estimated:

```
# combined output
mdl_R_out <- cbind(beta_est, test_res)
#
print(round(mdl_R_out, 5))
```

```
##                  coef_est     t_stat p_val
## intercept         4.01018  326.51051 0e+00
## x1                0.15906  169.41570 0e+00
## log_x2           -3.00237 -502.84282 0e+00
## married           0.04664   12.03513 0e+00
## age_gr1           0.02473    4.77617 0e+00
## age_gr2          -0.14683   -8.62067 0e+00
## age_gr2_x1        0.05037   30.61970 0e+00
## married_age_gr1 -0.02679   -4.05482 5e-05
```

In other words, we want to test the hypothesis with two restrictions:

$$\begin{cases} H_0 & : \beta_8 = 0, \ \beta_9 = 0 \\ H_1 & : \beta_8 \neq 0, \quad \text{or} \quad \beta_9 \neq 0, \quad \text{or both} \end{cases}$$

So, we can calculate the unrestricted and restricted residual sum of squares and the F-statistic along with the associated p-value:

```
RSS_UR <- sum(resid_UR^2)
RSS_R  <- sum(resid^2)
# F-statistic and its p-value
F_stat <- ((RSS_R - RSS_UR) / 2) / (RSS_UR / (nrow(data_mat) - length(beta_est_UR)))
p_val_F<- pf(F_stat, df1 = 2, df2 = nrow(data_mat) - length(beta_est_UR),
             lower.tail = FALSE)
#
print(cbind(F_stat, p_val_F))
```

```
##        F_stat   p_val_F
## [1,] 1.366289 0.2555323
```

In this case, the p-value is greater than 0.05, so we do not reject the null hypothesis that both coefficients are not statistically significantly different from zero.

Note that if we were to reject the null hypothesis - it would mean that **at least one** coefficient is statistically significantly different from zero - however, we would not know which one, or if both of the coefficients are significant.

There are a lot of important steps, from the design matrix creation when specifying the model, to the standard error and t-statistic calculation - Alternatively, we can specify the models with the built-in OLS estimation functions and carry out an **ANOVA** (Analysis of variance) test:

```
lm_UR <- lm(log(y) ~ x1 + log(x2) + married + age_gr1 + age_gr2 +
              age_gr2 * x1 + married * age_gr1 + married * age_gr2 +
              age_gr1 * x1, data = data_mat)
lm_R <- lm(log(y) ~ x1 + log(x2) + married + age_gr1 + age_gr2 +
             age_gr2 * x1 + married * age_gr1, data = data_mat)
#
print(anova(lm_R, lm_UR))
```

```
## Analysis of Variance Table
##
## Model 1: log(y) ~ x1 + log(x2) + married + age_gr1 + age_gr2 + age_gr2 *
##     x1 + married * age_gr1
## Model 2: log(y) ~ x1 + log(x2) + married + age_gr1 + age_gr2 + age_gr2 *
##     x1 + married * age_gr1 + married * age_gr2 + age_gr1 * x1
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    992 2.4314
## 2    990 2.4247  2 0.0066927 1.3663 0.2555
```

which give us the same F-statistic and p-values.

Testing for a Single Linear Restriction

Suppose that we are interested in testing the hypothesis that a linear combination of parameters:

$$\sum_{j=0}^{k} c_j \widehat{\beta}_j = c_0 \widehat{\beta}_0 + ... + c_k \widehat{\beta}_k = \widehat{r}$$

is equal to r:

$$\begin{cases} H_0 & : \widehat{r} = r \\ H_1 & : \widehat{r} \neq r \end{cases}$$

Then, the associated t-statistic is calculated by:

$$t_r = \frac{\widehat{r} - r}{\text{s.e.}(\widehat{r})} = \frac{\sum_{j=0}^{k} c_j \widehat{\beta}_j - \sum_{j=0}^{k} c_j \beta_j}{\text{s.e.} \left(\sum_{j=0}^{k} c_j \widehat{\beta}_j \right)} \sim t_{(N-(k+1))}$$

where $\text{s.e.} \left(\sum_{j=0}^{k} c_j \widehat{\beta}_j \right) = \sqrt{\widehat{\mathbb{V}\text{ar}} \left(\sum_{j=0}^{k} c_j \widehat{\beta}_j \right)}$, and

$\widehat{\mathbb{V}\text{ar}} \left(\sum_{j=0}^{k} c_j \widehat{\beta}_j \right) = \sum_{j=0}^{k} c_j^2 \cdot \widehat{\mathbb{V}\text{ar}} \left(\widehat{\beta}_j \right) + 2 \cdot \sum_{i<j} c_i c_j \cdot \widehat{\mathbb{C}\text{ov}} \left(\widehat{\beta}_i, \ \widehat{\beta}_j \right)$ Note that we can get the relevant values from the variance-covariance matrix estimate, $\widehat{\mathbb{V}\text{ar}}(\widehat{\boldsymbol{\beta}}) = \widehat{\sigma}^2 \left(\mathbf{X}^\top \mathbf{X} \right)^{-1}$.

If the sample size is large then, the errors will be approximately normally distributed.

Since the t-statistic has the same distribution as when testing for a single parameter, we can use the equivalent t_c values when testing either one-tail or two-tail hypothesis.

Note, that testing this linear constraint is **equivalent** to testing the following constraint on the parameter vector:

$$\mathbf{L}\widehat{\boldsymbol{\beta}} = \begin{bmatrix} c_0 & c_1 & \dots & c_k \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix} = r$$

We will talk about linear constraints a bit further on.

Example

We may be interested in testing whether one coefficient is eight times the magnitude of another coefficient. For example, if our null hypothesis is:

> A unit increase in X_1 (for a person from AGE_GROUP$_{OTHER}$) has an 8 times larger effect on the change in Y as the fact that a person is between 20 and 30 years old i.e. from AGE_GROUP$_1$

which can be written as the following hypothesis:

$$\begin{cases} H_0 & : \beta_1 = 8 \cdot \beta_4 \\ H_1 & : \beta_1 \neq 8 \cdot \beta_4 \end{cases} \iff \begin{cases} H_0 & : \beta_1 - 8 \cdot \beta_4 = 0 \\ H_1 & : \beta_1 - 8 \cdot \beta_4 \neq 0 \end{cases}$$

Then, our test t-statistic is:

$$t = \frac{\beta_1 - 8 \cdot \beta_4 - 0}{\text{s.e.}\left(\beta_1 - 8 \cdot \beta_4\right)} = \frac{\beta_1 - 8 \cdot \beta_4}{\text{s.e.}\left(\beta_1 - 8 \cdot \beta_4\right)}$$

We can then calculate the critical value t_c and test the hypothesis as we would for the single parameter case.

```r
# The variance of the linear restriction:
var_restr <- sum(diag(beta_vcov)[c(2, 5)] * c(1, -8)^2) + 2 * 1 * (-8) * beta_vcov[2, 5]
# Intermediate output:
print(paste0("beta[1] - 8 * beta[4] = ", round(beta_est[2] - 8 * beta_est[5], 5)))
```

```
## [1] "beta[1] - 8 * beta[4] = -0.03881"
```

```r
print(paste0("The s.e. of the difference = ", round(sqrt(var_restr), 5)))
```

```
## [1] "The s.e. of the difference = 0.04145"
```

```r
# The t-statistic of the linear restriction:
t_stat_restr <- (beta_est[2] - 8 * beta_est[5] - 0) / sqrt(var_restr)
# The associated p-value:
p_val_restr <- 2 * pt(-abs(t_stat_restr), df = nrow(data_mat) - length(beta_est), lower = TRUE)
#
print(cbind(t_stat_restr, p_val_restr))
```

```
##      t_stat_restr p_val_restr
## [1,]   -0.9361572   0.3494201
```

So, we do not reject the null hypothesis and conclude that *the estimated coefficient* $\widehat{\beta}_1$ *(i.e. the coefficient of* X_1*) is eight times larger than the estimated coefficient* $\widehat{\beta}_4$ *(i.e. the coefficient of AGE_GROUP*$_1$*).*

Thankfully, we have can automatically carry out this test:

```
lin_rest <- multcomp::glht(lm_R, linfct = c("x1 - 8 * age_gr1 = 0"))
print(summary(lin_rest))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = log(y) ~ x1 + log(x2) + married + age_gr1 + age_gr2 +
##     age_gr2 * x1 + married * age_gr1, data = data_mat)
##
## Linear Hypotheses:
##                        Estimate Std. Error t value Pr(>|t|)
## x1 - 8 * age_gr1 == 0 -0.03881    0.04145  -0.936    0.349
## (Adjusted p values reported -- single-step method)
```

We note that the values are identical to our manual calculation.
While in general, there may be at least one package available for most types of tests, this may not be the case across multiple different software.
In such cases, it is good to have an example of *manual* calculation across multiple software and have a built-in method in at least one of them for checking.

Example

Additionally, we want to test the following hypothesis that we have tested before:

$$\begin{cases} H_0 & : \beta_2 = -3 \\ H_1 & : \beta_2 \neq -3 \end{cases}$$

But this time, we formulate it as a linear restriction.

```
# We need to specify the formula WITHOUT any logarithm transformations to avoid possible errors:
lm_ht <- lm(log(y) ~ x1 + log_x2 + married + age_gr1 + age_gr2 + age_gr2 * x1 + married * age_gr1,
            data = data.frame(y = data_mat$y, x_mat_1))
#
summary(multcomp::glht(lm_ht, linfct = c("log_x2 = -3")))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = log(y) ~ x1 + log_x2 + married + age_gr1 + age_gr2 +
##     age_gr2 * x1 + married * age_gr1, data = data.frame(y = data_mat$y,
##     x_mat_1))
##
## Linear Hypotheses:
##                Estimate Std. Error t value Pr(>|t|)
## log_x2 == -3 -3.002369   0.005971  -0.397    0.692
## (Adjusted p values reported -- single-step method)
```

Since the $p - value > 0.05$, we have no grounds to reject the null hypothesis and conclude that β_2 is not statistically significantly different from -3.

Note that this is identical to what we have manually calculated:

```
print(hyp_b2)
```

```
##            t_stat     p_val
## log_x2 -0.3967519 0.6916357
```

We note that in R, using `log(x2)` instead of `log_x2` produces an error (though this may be different for different package versions). So knowing how to manually calculate this hypothesis test is much more useful than simply applying some black-box functions:

```
lm_ht_v2 <- lm(log(y) ~ x1 + log(x2) + married + age_gr1 + age_gr2 +
                 age_gr2 * x1 + married * age_gr1,
               data = data.frame(y = data_mat$y, x_mat_1))
#
summary(multcomp::glht(lm_ht_v2, linfct = c("log(x2) = -3")))
```

```
## Error: multcomp:::expression2coef::walkCode::eval: the expression 'log(x2)' did not evaluate to a
```

Example

We want to test the following hypothesis that one coefficient is two times larger than the other:

$$\begin{cases} H_0 & : \beta_1 = 2 \cdot \beta_6 \\ H_1 & : \beta_1 \neq 2 \cdot \beta_6 \end{cases}$$

```
# Print the coefficient names:
names(coef(lm_R))
```

```
## [1] "(Intercept)"     "x1"              "log(x2)"         "married"
## [5] "age_gr1"         "age_gr2"         "x1:age_gr2"      "married:age_gr1"
```

```
# Specify the linear restriction
lin_rest <- multcomp::glht(lm_R, linfct = c("x1 - 2 * x1:age_gr2 = 0"))
print(summary(lin_rest))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = log(y) ~ x1 + log(x2) + married + age_gr1 + age_gr2 +
##     age_gr2 * x1 + married * age_gr1, data = data_mat)
##
## Linear Hypotheses:
##                            Estimate Std. Error t value Pr(>|t|)
## x1 - 2 * x1:age_gr2 == 0 0.058333   0.003902   14.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

In this case, we would reject the null hypothesis and conclude that the coefficients are different by a magnitude, which is either greater, or lower than 2.

On the other hand, we know that the true parameter values are $\beta_1 = 0.16$ and $\beta_6 = 0.05$, so the true magnitude is 3.2. If we specify the hypothesis with the true magnitude:

$$\begin{cases} H_0 & : \beta_1 = 3.2 \cdot \beta_6 \\ H_1 & : \beta_1 \neq 3.2 \cdot \beta_6 \end{cases}$$

then:

```r
# Specify the linear restriction
lin_rest <- multcomp::glht(lm_R, linfct = c("x1 - 3.2 * x1:age_gr2 = 0"))
print(summary(lin_rest))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = log(y) ~ x1 + log(x2) + married + age_gr1 + age_gr2 +
##     age_gr2 * x1 + married * age_gr1, data = data_mat)
##
## Linear Hypotheses:
##                            Estimate Std. Error t value Pr(>|t|)
## x1 - 3.2 * x1:age_gr2 == 0 -0.002106   0.005850   -0.36    0.719
## (Adjusted p values reported -- single-step method)
```

We would not reject the null hypothesis that β_1 is 3.2 times larger than β_6.

Takeaway

If we were to estimate a model of the effects of *price* and *advertising* expenditure (in dollars) on the *revenue*. In such a case β_{price} would be the change in revenue from a one dollar increase in price, and $\beta_{advertising}$ would be the change in revenue from a one dollar increase in advertising. So, one Hypothesis could be formulated as:

> reducing the price by 10 cents would have the same effect on revenue as increasing the advertising by 100 dollars,

which would translate to the following hypothesis: $H_0 : -0.1\beta_{price} = 100\beta_{advertising}$. The alternative could be that $H_1 : -0.1\beta_{price} > 100\beta_{advertising}$, i.e.:

> reducing the price by 10 cents would be more effective than increasing the advertising by 100 dollars.

Finally, it is usually more common to use use the tests provided further on, as they can be used for more than one linear restriction.

In our first example, under the null hypothesis we have that our regression could be re-written as the following **restricted** regression:

$$\log(Y_i) = \beta_0 + \beta_1 (X_{1i} + 8 \cdot \text{AGE\_GROUP}_{1i}) + \beta_2 \log(X_{2i}) + \beta_3 \text{MARRIED}_i + \beta_5 \text{AGE\_GROUP}_{2i} + \beta_6(\text{AGE\_GROUP}_{2i} \times X_{1i}) + \beta_7(\text{MARRIED}_i \times \text{AGE\_GROUP}_{1i}) + \epsilon_i$$

Then, we could compare the residual sum of squares from this model with the ones from the unrestricted regression (i.e. the regression under the alternative) via the F-test.

Testing for Multiple Linear Restrictions

If we want to test $M < k + 1$ different linear restriction on the coefficients, then we can define $\mathbf{L}$ and the value vector of the linear restrictions as:

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

We want to test the following hypothesis:

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

The distribution of $\mathbf{L}\widehat{\boldsymbol{\beta}}$ is:

$$\mathbf{L}\widehat{\boldsymbol{\beta}} \sim \mathcal{N}\left(\mathbf{L}\boldsymbol{\beta},\ \mathbf{L}\mathbb{V}\mathrm{ar}(\widehat{\boldsymbol{\beta}})\mathbf{L}^\top\right)$$

where:

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

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

Where the variances of the linearly restricted parameters are the diagonal elements:

$$\begin{aligned}\mathrm{diag}\left(\mathbf{L}\mathbb{V}\mathrm{ar}(\widehat{\boldsymbol{\beta}})\mathbf{L}^\top\right) &= \left\{\sum_{j=0}^{k} c_{i,j}^2 \cdot \mathbb{V}\mathrm{ar}\left(\widehat{\beta}_j\right) + \sum_{j_1\neq j_2} c_{i,j_1}c_{i,j_2}\cdot\mathbb{C}\mathrm{ov}\left(\widehat{\beta}_{j_1},\ \widehat{\beta}_{j_2}\right)\right\}_{i=1,\ldots,M} \\ &= \left\{\sum_{j=0}^{k} c_{i,j}^2 \cdot \mathbb{V}\mathrm{ar}\left(\widehat{\beta}_j\right) + 2\cdot\sum_{j_1<j_2} c_{i,j_1}c_{i,j_2}\cdot\mathbb{C}\mathrm{ov}\left(\widehat{\beta}_{j_1},\ \widehat{\beta}_{j_2}\right)\right\}_{i=1,\ldots,M}\end{aligned}$$

In practice, we replace σ^2 with $\widehat{\sigma}^2$.

Since we have more than one restriction, a t-**test is not applicable**. Nevertheless, there are a number of alternative test statistics, that we can use

Wald test for Multiple Linear Restrictions

One can calculate the **Wald test** statistic, which is applicable to **large samples**:

$$W = \left(\mathbf{L}\widehat{\boldsymbol{\beta}} - \boldsymbol{r}\right)^{\top} \left[\mathbf{L}\mathbb{V}\mathrm{ar}(\widehat{\boldsymbol{\beta}})\mathbf{L}^{\top}\right]^{-1} \left(\mathbf{L}\widehat{\boldsymbol{\beta}} - \boldsymbol{r}\right) \sim \chi^2_M$$

Note that we do not know the true $\mathbb{V}\mathrm{ar}(\widehat{\boldsymbol{\beta}})$. If the sample size is large, then the estimated $\widehat{\sigma}^2$ is close to the true population variance and the Wald test may be applicable.

F-test for Multiple Linear Restrictions

In practice, if we replace σ^2 with $\widehat{\sigma}^2$, and divide the statistic by the number of restrictions, M - which is applicable for smaller samples - we get the following $F - \textit{statistic}$:

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

Alternatively:

$$F = \frac{(\text{RSS}_R - \text{RSS}_{UR})/M}{\text{RSS}_{UR}/(N-(k+1))} = \frac{(R^2_{UR} - R^2_R)/M}{(1-R^2_{UR})/(N-(k+1))} \sim F_{(M,N-(k+1))}$$

where RSS_R is the residual sum of squares of the restricted model (i.e. under the null hypothesis), and RSS_{UR} is the residual sum of squares of the unrestricted model (i.e. under the alternative hypothesis).

Take note that, regardless of the restrictions, the R-squared must still be calculated for the same dependent variable. E.g. setting a restriction that $\beta_j = 1$ would allow us to create a restricted model on $Y_i - X_{j,i}$, instead of Y_i.

This may be software-dependent, so it may sometimes be a good idea to examine the fitted-values, or compare to manually calculated estimates, to make sure that the same dependent variable is estimated in both cases.

Regardless of the chosen formula, we then need to calculate the relevant $F_c = F_{(1-\alpha,M,N-(k+1))}$ and the associated p-value.

Example

If we have the following multiple regression:

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{3i} + \beta_4 X_{4i} + \beta_5 X_{i5} + \epsilon_i$$

We may be interested in testing if two pairs of parameters have the same pair-wise effect on Y:

$$\begin{cases} H_0 & : \beta_2 = \beta_3, \quad \beta_4 = 2 \cdot \beta_5 \\ H_1 & : \beta_2 - \beta_3 \neq 0 \quad \text{or} \quad \beta_4 - 2 \cdot \beta_5 \neq 0 \quad \text{or both} \end{cases}$$

(Note: we have written the alternative hypothesis differently to highlight how we are going to specify the **L** matrix).
In this case, our constraint matrix and the associated value vector is:

$$\mathbf{L} = \begin{bmatrix} 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 \end{bmatrix}, \quad \boldsymbol{r} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Example

The joint hypothesis for model parameter significance hypothesis testing is equivalent to k different restrictions:

$$\begin{cases} H_0 & : \beta_1 = ... = \beta_k = 0 \\ H_1 & : \beta_j \neq 0, \text{ for some } j \end{cases}$$

Our constraint matrix and the associated value vector is:

$$\mathbf{L} = \begin{bmatrix} 0 & 1 & 0 & 0 & ... & 0 \\ 0 & 0 & 1 & 0 & ... & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & ... & 1 \end{bmatrix}, \quad \boldsymbol{r} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Note that this is equivalent to the joint hypothesis test for multiple coefficient significance.

What happens if we cannot reject the null hypothesis? This means that there is a significant linear combination of some of our model parameters.
We would like to try to incorporate this information into our coefficient estimator. We can do this by carrying out a **Restricted Least Squares** (or **Constrained Least Squares**) procedure.
If we ignore this information, then the OLS estimates are still **unbiased** but not as effective as RLS.
On the other hand, how do we even know what kind of restrictions we should be testing for? Generally, we may sometimes want to impose (close-to-)zero-value restrictions on some specific coefficients, which we cannot easily remove from the model specification.

Example

We can test the following hypothesis with an F-test, instead of a t-test:

$$\begin{cases} H_0 & : \beta_2 = -3 \\ H_1 & : \beta_2 \neq -3 \end{cases}$$

```
car::linearHypothesis(lm_R, "log(x2) = -3")
```

```
## Linear hypothesis test
##
## Hypothesis:
## log(x2) = - 3
##
## Model 1: restricted model
## Model 2: log(y) ~ x1 + log(x2) + married + age_gr1 + age_gr2 + age_gr2 *
##     x1 + married * age_gr1
##
##   Res.Df    RSS Df  Sum of Sq      F Pr(>F)
## 1    993 2.4318
## 2    992 2.4314  1 0.00038582 0.1574 0.6916
```

The advantage of the t-test is that we can do it directly from the usual regression output, even if we weren't sure whether we would need to perform any hypothesis testing.

The relationship between a t-statistic and an F-statistic that has **one degree of freedom in the numerator** (i.e. one restriction) is:

$$F_{(1,N-(k+1))} = t^2_{(N-(k+1))}$$

If we look at the squared t-statistic from our previous t-test:

```
print(summary(multcomp::glht(lm_ht, linfct = c("log_x2 = -3")))$test$tstat^2)
```

```
##    log_x2
## 0.1574121
```

we see that it is the same as the F-statistic from our F-test.

Example

Say we want to test two linear restrictions:

$$\begin{cases} H_0 & : \beta_2 = -3,\ \beta_3 + \beta_6 = 0.09, \\ H_1 & : \beta_2 \neq -3,\ \text{or } \beta_3 + \beta_6 \neq 0.09,\ \text{or both} \end{cases}$$

We will first show how to carry out this test manually. We begin by specifying the linear restriction matrix and the value vector for $M = 2$ restrictions:

$$\mathbf{L} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad \boldsymbol{r} = \begin{bmatrix} -3 \\ 0.09 \end{bmatrix}$$

```
L_mat <- matrix(0, nrow = 2, ncol = length(beta_est))
L_mat[1, 3] <- 1
L_mat[2, c(4, 7)] <- 1
r_vec <- c(-3, 0.09)
#
print(L_mat)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    0    0    1    0    0    0    0    0
## [2,]    0    0    0    1    0    0    1    0
```

```
print(r_vec)
```

```
## [1] -3.00  0.09
```

Next, since we have already estimated the variance-covariance matrix of our `beta_est` variable, we can calculate the F-statistic:

```
M <- 2
F_stat_mult_lin <- 1 / M * t(L_mat %*% beta_est - r_vec)
F_stat_mult_lin <- F_stat_mult_lin %*% solve(L_mat %*% beta_vcov %*% t(L_mat))
F_stat_mult_lin <- F_stat_mult_lin %*% (L_mat %*% beta_est - r_vec)
F_stat_mult_lin <- c(F_stat_mult_lin)
#
p_val_F_mult_lin <- pf(F_stat_mult_lin,
                       df1 = M, df2 = nrow(data_mat) - length(beta_est_UR),
                       lower.tail = FALSE)
print(cbind(F_stat_mult_lin, p_val_F_mult_lin))
```

```
##      F_stat_mult_lin p_val_F_mult_lin
## [1,]        1.474699        0.2293498
```

We can also do this automatically:

```
car::linearHypothesis(lm_R, c("log(x2) = -3", "married + x1:age_gr2 = 0.09"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## log(x2) = - 3
## married  + x1:age_gr2 = 0.09
##
## Model 1: restricted model
## Model 2: log(y) ~ x1 + log(x2) + married + age_gr1 + age_gr2 + age_gr2 *
##     x1 + married * age_gr1
##
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    994 2.4386
## 2    992 2.4314  2 0.0072291 1.4747 0.2293
```

We have no grounds to reject the null hypothesis ($p - value > 0.05$).

We can also carry out the Wald test:

```
Wald_stat = F_stat_mult_lin * M
#
p_val_Wald <- pchisq(Wald_stat, df = M, lower.tail = FALSE)
#
print(cbind(Wald_stat, p_val_Wald))
```

```
##      Wald_stat p_val_Wald
## [1,]  2.949399  0.2288475
```

and we can compare it to the built-in functions:

```
car::linearHypothesis(lm_R, c("log(x2) = -3", "married + x1:age_gr2 = 0.09"), test = "Chisq")
```

```
## Linear hypothesis test
##
## Hypothesis:
## log(x2) = - 3
## married  + x1:age_gr2 = 0.09
##
## Model 1: restricted model
## Model 2: log(y) ~ x1 + log(x2) + married + age_gr1 + age_gr2 + age_gr2 *
##     x1 + married * age_gr1
##
##   Res.Df    RSS Df Sum of Sq  Chisq Pr(>Chisq)
## 1    994 2.4386
## 2    992 2.4314  2 0.0072291 2.9494     0.2288
```

In this example, the Wald test results give the same conclusions as the F-test.

Built-in Estimation Functions

We have seen a brief example on how to carry out multiple regression OLS using the built-in functions - now, we will provide them separately. The aim is to highlight that these functions produce the same results, as the manual approach (i.e. by applying the previously discussed formulas):

Parameter OLS Estimation, Significance Testing

```
mdl <- lm(log(y) ~ x1 + log(x2) + married + age_gr1 + age_gr2 + age_gr2 * x1 + married * age_gr1,
          data = data_mat)
print(summary(mdl))
```

```
##
## Call:
## lm(formula = log(y) ~ x1 + log(x2) + married + age_gr1 + age_gr2 +
##     age_gr2 * x1 + married * age_gr1, data = data_mat)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.129248 -0.034729  0.001534  0.034763  0.163707
##
## Coefficients:
##                    Estimate Std. Error  t value Pr(>|t|)
## (Intercept)       4.0101765  0.0122819  326.511  < 2e-16 ***
## x1                0.1590640  0.0009389  169.416  < 2e-16 ***
## log(x2)          -3.0023689  0.0059708 -502.843  < 2e-16 ***
## married           0.0466433  0.0038756   12.035  < 2e-16 ***
## age_gr1           0.0247338  0.0051786    4.776 2.06e-06 ***
## age_gr2          -0.1468251  0.0170318   -8.621  < 2e-16 ***
## x1:age_gr2        0.0503657  0.0016449   30.620  < 2e-16 ***
## married:age_gr1  -0.0267870  0.0066062   -4.055 5.41e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04951 on 992 degrees of freedom
## Multiple R-squared:  0.997,  Adjusted R-squared:  0.9969
## F-statistic: 4.647e+04 on 7 and 992 DF,  p-value: < 2.2e-16
```

We can also extract the variance-covariance matrix of the parameters:

```
print(vcov(mdl))
```

```
##                   (Intercept)            x1       log(x2)       married
## (Intercept)      1.508456e-04 -8.758739e-06 -4.316298e-05 -9.329513e-06
## x1              -8.758739e-06  8.815287e-07  7.508965e-09  1.431190e-07
## log(x2)         -4.316298e-05  7.508965e-09  3.565033e-05 -1.114429e-07
## married         -9.329513e-06  1.431190e-07 -1.114429e-07  1.502027e-05
## age_gr1         -9.713703e-06 -7.153594e-08 -1.090457e-06  8.035502e-06
## age_gr2         -8.840137e-05  8.664538e-06 -3.808553e-06  6.299750e-07
## x1:age_gr2       8.300376e-06 -8.803845e-07  3.122693e-07 -3.112915e-08
## married:age_gr1  5.680298e-06  5.500742e-08  1.503668e-06 -1.499250e-05
##                       age_gr1       age_gr2    x1:age_gr2 married:age_gr1
## (Intercept)     -9.713703e-06 -8.840137e-05  8.300376e-06    5.680298e-06
## x1              -7.153594e-08  8.664538e-06 -8.803845e-07    5.500742e-08
## log(x2)         -1.090457e-06 -3.808553e-06  3.122693e-07    1.503668e-06
## married          8.035502e-06  6.299750e-07 -3.112915e-08   -1.499250e-05
## age_gr1          2.681785e-05  6.419738e-06  1.221773e-07   -2.313111e-05
## age_gr2          6.419738e-06  2.900805e-04 -2.727918e-05    1.165977e-06
## x1:age_gr2       1.221773e-07 -2.727918e-05  2.705623e-06   -1.542676e-07
## married:age_gr1 -2.313111e-05  1.165977e-06 -1.542676e-07    4.364213e-05
```

We can compare it to the manually calculated variance-covariance matrix:

```
# manually calculatd matrix
print(beta_vcov)
```

```
##                     intercept            x1        log_x2       married
## intercept        1.508456e-04 -8.758739e-06 -4.316298e-05 -9.329513e-06
## x1              -8.758739e-06  8.815287e-07  7.508965e-09  1.431190e-07
## log_x2          -4.316298e-05  7.508965e-09  3.565033e-05 -1.114429e-07
## married         -9.329513e-06  1.431190e-07 -1.114429e-07  1.502027e-05
## age_gr1         -9.713703e-06 -7.153594e-08 -1.090457e-06  8.035502e-06
## age_gr2         -8.840137e-05  8.664538e-06 -3.808553e-06  6.299750e-07
## age_gr2_x1       8.300376e-06 -8.803845e-07  3.122693e-07 -3.112915e-08
## married_age_gr1  5.680298e-06  5.500742e-08  1.503668e-06 -1.499250e-05
##                      age_gr1       age_gr2    age_gr2_x1 married_age_gr1
## intercept       -9.713703e-06 -8.840137e-05  8.300376e-06    5.680298e-06
## x1              -7.153594e-08  8.664538e-06 -8.803845e-07    5.500742e-08
## log_x2          -1.090457e-06 -3.808553e-06  3.122693e-07    1.503668e-06
## married          8.035502e-06  6.299750e-07 -3.112915e-08   -1.499250e-05
## age_gr1          2.681785e-05  6.419738e-06  1.221773e-07   -2.313111e-05
## age_gr2          6.419738e-06  2.900805e-04 -2.727918e-05    1.165977e-06
## age_gr2_x1       1.221773e-07 -2.727918e-05  2.705623e-06   -1.542676e-07
## married_age_gr1 -2.313111e-05  1.165977e-06 -1.542676e-07    4.364213e-05
```

We can compare the relevant values:

- the estimated coefficients and their standard errors,
- the corresponding t-value for the null hypothesis tests,
- the residual standard error,
- degrees of freedom
- the F-statistic for the hypothesis that all the slope coefficients (i.e. excluding β_0) are not statistically significantly different from zero, against the alternative that at least one is statistically significantly different.

Categorical Data Handling

Additionally, we have a **categorical variable** in our `data_mat` matrix - it is the `age_group` variable. Instead of manually calculating the dummy indicator variables, we can directly include it in the model:

```r
# Specify the model with categorical variable and interaction terms:
mdl2 <- lm(log(y) ~ x1 + log(x2) + married + age_group + age_group * x1 + married * age_group,
           data = data_mat)
print(summary(mdl2))
```

```
##
## Call:
## lm(formula = log(y) ~ x1 + log(x2) + married + age_group + age_group *
##     x1 + married * age_group, data = data_mat)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.130645 -0.033463  0.001745  0.034883  0.159463
##
## Coefficients:
##                                  Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                      4.025357   0.015556  258.762  < 2e-16 ***
## x1                               0.157716   0.001321  119.403  < 2e-16 ***
## log(x2)                         -3.002311   0.005969 -502.990  < 2e-16 ***
## married                          0.042931   0.005480    7.834 1.21e-14 ***
## age_groupaged_20_30             -0.003679   0.019615   -0.188  0.85125
## age_groupaged_31_65             -0.164027   0.019968   -8.215 6.61e-16 ***
## x1:age_groupaged_20_30           0.002663   0.001879    1.417  0.15668
## x1:age_groupaged_31_65           0.051739   0.001890   27.378  < 2e-16 ***
## married:age_groupaged_20_30     -0.022777   0.007664   -2.972  0.00303 **
## married:age_groupaged_31_65      0.007016   0.007754    0.905  0.36578
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

On the other hand, if we want to remove insignificant interaction terms and keep only the significant ones - we will need to use our indicator variables:

```
# Specify the model with categorical variable and interaction terms:
mdl3 <- lm(log(y) ~ x1 + log(x2) + married + age_group + age_gr2:x1 + married:age_gr1,
           data = data_mat)
print(summary(mdl3))
```

```
##
## Call:
## lm(formula = log(y) ~ x1 + log(x2) + married + age_group + age_gr2:x1 +
##     married:age_gr1, data = data_mat)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.129248 -0.034729  0.001534  0.034763  0.163707
##
## Coefficients:
##                          Estimate Std. Error  t value Pr(>|t|)
## (Intercept)             4.0101765  0.0122819  326.511  < 2e-16 ***
## x1                      0.1590640  0.0009389  169.416  < 2e-16 ***
## log(x2)                -3.0023689  0.0059708 -502.843  < 2e-16 ***
## married                 0.0466433  0.0038756   12.035  < 2e-16 ***
## age_groupaged_20_30     0.0247338  0.0051786    4.776 2.06e-06 ***
## age_groupaged_31_65    -0.1468251  0.0170318   -8.621  < 2e-16 ***
## x1:age_gr2              0.0503657  0.0016449   30.620  < 2e-16 ***
## married:age_gr1        -0.0267870  0.0066062   -4.055 5.41e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04951 on 992 degrees of freedom
## Multiple R-squared:  0.997,  Adjusted R-squared:  0.9969
```

Consequences of removing insignificant indicator variable interaction terms

Generally, if some interaction terms are insignificant as additional explanatory variables, removing them from the model changes the interpretation of the **base category**. For example, in the following model:

$$\log(Y_i) = \beta_0 + \beta_1 X_{1i} + \beta_2 \log(X_{2i}) + \beta_3 \text{AGE\_GROUP}_{1i} + \beta_4 \text{AGE\_GROUP}_{2i} + \beta_5 (\text{AGE\_GROUP}_{2i} \times X_{1i}) + \epsilon_i$$

since our base age category is AGE_GROUP_{OTHER}, excluding the interaction term $(\text{AGE\_GROUP}_{1i} \times X_{1i})$, implies that the coefficient β_5 of the interaction variable now compares to a different base group: $\text{AGE\_GROUP}_{OTHER} + \text{AGE\_GROUP}_1$.
In other words:

- coefficients β_3 and β_4 are the effects of different age groups compared to the AGE_GROUP_{OTHER} group;
- coefficient β_5 is the additional effect of a unit increase in X_1 from people in AGE_GROUP_2, **compared to the remaining two groups** $\text{AGE\_GROUP}_{OTHER} + \text{AGE\_GROUP}_1$.

If we were to leave the interaction variable $(\text{AGE\_GROUP}_{1i} \times X_{1i})$, then β_5 would be the additional effect of a unit increase in X_1 from people in AGE_GROUP_1, compared to the base age group AGE_GROUP_{OTHER}.

For this reason, if there are many insignificant interaction terms, or insignificant category levels, we need to decide on the following:

- Re-categorize the data - combine some categories together, or combine insignificant categories into a new, `other`, category, as long as that grouping **makes economic sense**.
- Leave the groups as they are, along with insignificant interaction terms and/or insignificant category levels. This makes interpretation consistent, since we will have **the same base group for individual category effects and interaction effects**.

Finally, inclusion of MARRIED indicator variable, indicating whether a person is married, would compare to the base group of `unmarried` people, **regardless of their age**. Consequently, the base age group AGE_GROUP$_{OTHER}$ ignores whether a person is single or married - it only indicates their age group.

In other words, the categorical variables MARRIED (with its two categories) and AGE_GROUP (with its three categories) will have base groups, that are not necessarily interpreted the same - marriage status does not take into account age, and age does not take into account marriage status.

Goodness-Of-Fit

Let our multiple regression model be defined as:

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

Furthermore, assume that conditions (MR.1) - (MR.6) hold true.

F-test For Goodness of Fit: Joint Hypothesis Test for the Overall Model Significance

This is the joint hypothesis test for multiple coefficient significance, applied for **all slope coefficients** (*excluding the intercept*):

$$\begin{cases} H_0 & : \beta_1 = \beta_2 = ... = \beta_k = 0 \\ H_1 & : \beta_j \neq 0, \quad \text{for some } j \end{cases}$$

the associated F-statistic:

$$F = \frac{\text{ESS}/k}{\text{RSS}/(N-(k+1))} = \frac{R^2/k}{(1-R^2)/(N-(k+1))}$$

If $F > F_{(1-\alpha,k,N-(k+1))}$ for some significance level α (or if the associated p-value is less than α), we reject the null hypothesis.

- The F-test of overall significance compares a model with no predictors to the specified model.
- A regression model that contains no predictors is also known as an intercept-only model, where all of the fitted values equal the mean of the response variable.
- Thus, if the p-value of the overall F-test is less than the significance level α, the specified model predicts the response variable better than the mean of the response.

This is somewhat similar to what the R^2 does. However, it does not provide a formal hypothesis test (which the overall F-test does).

Consequently, if the p-value of the overall F-test is less than the α significance level, we could conclude that the R^2 value is significantly different from zero.

R-squared and Adjusted R-squared

In the multiple regression model R^2 is a measure of the proportion of variation in the dependent variable, that is explained by **all** the explanatory variables included in the model:

$$R^2 = \frac{\text{ESS}}{\text{TSS}} = \frac{\sum_{i=1}^{N}\left(\widehat{Y}_i - \overline{Y}\right)^2}{\sum_{i=1}^{N}\left(Y_i - \overline{Y}\right)^2} = 1 - \frac{\text{RSS}}{TSS} = 1 - \frac{\sum_{i=1}^{N}\widehat{\epsilon}_i^2}{\sum_{i=1}^{N}\left(Y_i - \overline{Y}\right)^2}$$

where $\widehat{\epsilon}_i = Y_i - (\widehat{\beta}_0 + \widehat{\beta}_1 X_{1,i} + ... + \widehat{\beta}_k X_{k,i})$.

For the multiple regression model, R^2 automatically increases when extra explanatory variables are added to the model, even if the variables added have no justification: if the model contains $N - 1$ variables, then $R^2 = 1$.

As such, an **adjusted** R^2, R^2_{adj} may be used. This modification of R^2 adjusts for the number of explanatory variable (excluding the constant) terms in a model, relative to the number of data points. Furthermore, R^2_{adj} can be negative and $R^2_{adj} \leq R^2$. The adjusted R^2 is defined as:

$$R^2_{adj} = 1 - (1 - R^2)\frac{N-1}{N-k-1} = 1 - \frac{\text{RSS}/(N-k-1)}{\text{TSS}/(N-1)}$$

The adjusted R^2, R^2_{adj}, can be interpreted as an unbiased estimator of the population R^2.

However, because of this correction, R^2_{adj} loses its interpretation - R^2_{adj} is no longer the proportion of explained variation.

AIC and BIC

Selecting variables that maximize R^2_{adj} is equivalent to selecting variables that minimize ESS, subject to a penalty based on the number of variables. We will introduce two mode criterions, which work in a similar way, but have different penalties for inclusion of additional variables:

- **Akaike Information Criterion (AIC)**:

$$AIC = N + N\log(2\pi) + N\log\left(\frac{RSS}{N}\right) + 2(k+1)$$

- **Bayesian Information Criterion (BIC)**, also called **Schwarz Criterion (SC)**:

$$BIC = N + N\log(2\pi) + N\log\left(\frac{RSS}{N}\right) + (k+1)\log(N)$$

The formulas are based on Source.

- On one hand, the variance parameter (which is also estimated) must also be included: overall, there are $k+2$ total parameters $\beta_0, \beta_1, ..., \beta_k, \sigma^2$. As such $(k+1)$ is sometimes replaced with $(k+2)$.
- On the other hand, some textbooks ignore the first two terms, $N + N\log(2\pi)$, and use $k+1$, instead of $k+2$.

Note: in 'R' $(k+2)$ is used, while in 'Python' $(k+1)$ is used.

In both cases, the first term decreases with each extra variable added, but the second term increases, penalizing the inclusion of additional variables. **BIC** penalizes extra variables more strictly, compared to the **AIC**.

The model with the **smallest** AIC (or BIC) is preferred.

Example

We will use the following model to aid our presented methodology:

$$\log(Y_i) = \beta_0 + \beta_1 X_{1i} + \beta_2 \log(X_{2i}) + \beta_3 \text{MARRIED}_i + \beta_4 \text{AGE\_GROUP}_{1i} + \beta_5 \text{AGE\_GROUP}_{2i} + \beta_6(\text{AGE\_GROUP}_{2i} \times X_{1i}) + \beta_7(\text{MARRIED}_i \times \text{AGE\_GROUP}_{1i}) + \epsilon_i$$

where $MARRIED_i = 1$, if the i-th person is married, 0 otherwise; AGE_GROUP_{ji} are different age groups: if $j = 1$ - between $20 - 30$; if $j = 2$ - between $31 - 65$, the **base group**, AGE_GROUP_{OTHER}, consists the people with ages in the remaining age brackets, not covered by $j = 1, 2$.

We then generate the variables in the following way:

```
set.seed(132)
#
N <- 1000
beta_vec <- c(4, 0.16, -3, 0.05, 0.02, -0.15, 0.05, -0.03)
#
e <- rnorm(mean = 0, sd = 0.05, n = N)
x1<- rnorm(mean = 10, sd = 2, n = N)
x2<- sample(seq(from = 2, to = 5, length.out = floor(N * 0.8)),
            size = N, replace = TRUE)
#
married <- sample(c(0, 1), size = N, replace = TRUE)
```

The different age groups can be generated randomly as well. We can further create separate indicator variables for two of the three groups. Doing it this way automatically classifies the remaining group of other ages as the base group and we will avoid the dummy variable trap:

```
age_group <- sample(c("other", "aged_20_30", "aged_31_65"),
                    size = N, replace = TRUE)
age_gr1 <- rep(0, N)
age_gr1[age_group %in% "aged_20_30"] <- 1
age_gr2 <- rep(0, N)
age_gr2[age_group %in% "aged_31_65"] <- 1
```

Finally, we can create our dependent variable and combine all the data into a single dataset:

```
x_mat <- cbind(1, x1, log(x2), married,
               age_gr1, age_gr2, age_gr2 * x1 , married * age_gr1)
colnames(x_mat) <- c("intercept", "x1", "log_x2", "married",
                     "age_gr1", "age_gr2", "age_gr2_x1", "married_age_gr1")
#
y <- exp(x_mat %*% beta_vec + e)
#
data_mat <- data.frame(y, x1, x2, married, age_gr1, age_gr2, age_group)
head(data_mat)
```

```
##           y        x1       x2 married age_gr1 age_gr2  age_group
## 1 29.313052 10.206100 2.450563       1       0       1 aged_31_65
## 2 14.656237 10.164646 2.976220       0       0       1 aged_31_65
## 3 28.228720 10.816951 2.255319       1       0       0      other
## 4  7.741518 11.713026 4.286608       1       0       1 aged_31_65
## 5  9.333522  6.220738 2.514393       1       1       0 aged_20_30
## 6  2.165643  5.813722 4.237797       1       0       1 aged_31_65
```

Next, we estimate the models and print their output:

```
mdl <- lm(log(y) ~ x1 + log(x2) + married + age_gr1 + age_gr2 + age_gr2 * x1 + married * age_gr1,
          data = data_mat)
print(summary(mdl))
```

```
##
## Call:
## lm(formula = log(y) ~ x1 + log(x2) + married + age_gr1 + age_gr2 +
##     age_gr2 * x1 + married * age_gr1, data = data_mat)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.129248 -0.034729  0.001534  0.034763  0.163707
##
## Coefficients:
##                    Estimate Std. Error  t value Pr(>|t|)
## (Intercept)       4.0101765  0.0122819  326.511  < 2e-16 ***
## x1                0.1590640  0.0009389  169.416  < 2e-16 ***
## log(x2)          -3.0023689  0.0059708 -502.843  < 2e-16 ***
## married           0.0466433  0.0038756   12.035  < 2e-16 ***
## age_gr1           0.0247338  0.0051786    4.776 2.06e-06 ***
## age_gr2          -0.1468251  0.0170318   -8.621  < 2e-16 ***
## x1:age_gr2        0.0503657  0.0016449   30.620  < 2e-16 ***
## married:age_gr1 -0.0267870  0.0066062   -4.055 5.41e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04951 on 992 degrees of freedom
## Multiple R-squared:  0.997,  Adjusted R-squared:  0.9969
## F-statistic: 4.647e+04 on 7 and 992 DF,  p-value: < 2.2e-16
```

```r
print(AIC(mdl))
```

```
## [1] -3163.401
```

```r
print(BIC(mdl))
```

```
## [1] -3119.231
```

We also manually estimate AIC and BIC (again note that **the formulas in R are different from the ones in Python**):

```r
aic_lm <- nrow(mdl$model) + nrow(mdl$model) * log(2 * pi) +
  nrow(mdl$model) * log(sum(mdl$residual^2) / nrow(mdl$model)) +
  2 * (length(mdl$coefficients) + 1)
print(aic_lm)
```

```
## [1] -3163.401
```

```r
bic_lm <- nrow(mdl$model) + nrow(mdl$model) * log(2 * pi) +
  nrow(mdl$model) * log(sum(mdl$residual^2) / nrow(mdl$model)) +
  (length(mdl$coefficients) + 1) * log(nrow(mdl$model))
print(bic_lm)
```

```
## [1] -3119.231
```

Out-of-Sample (Hold-Out-Sample) GoF Testing

If our model is designed for prediction/forecasting, we want to evaluate its ability to forecast the dependent variable values, which have not been observed yet. One way to do this is to **hold-back** some of the observations from estimation and evaluate how well can the model predict the omitted observations (also known as splitting the data into an 80% *training* and a 20% *testing* set).

Lets say that out of the N observations, we use $N - m$ to estimate the parameters and then calculate the predictions:

$$\widehat{Y}_i = \widehat{\beta}_0 + \widehat{\beta}_1 X_{1,i} + ... + \widehat{\beta}_k X_{k,i}, \quad i = N - m + 1, ..., N$$

Then, we can measure the models **out-of-sample** forecasting accuracy by calculating the **Root Mean Squared Error (RMSE)**:

$$RMSE = \sqrt{\frac{1}{m} \sum_{i=N-m+1}^{N} \left(Y_i - \widehat{Y}_i \right)^2}$$

We can then compare different models based on their *RMSE* (as long as the models being compared have the same dependent variable, as well as the same amount of held-back observations, m).