Chapter 6

Multiple Linear Regression

Contents

In Chapter 5 we described the linear regression model, when the outcome (Y) is a linear function of *one* regressor (x). It is natural to extend this model to include more than one regressor, in general we can write

$$Y_i = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_p x_{p,i} + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2), \tag{6-1}$$

where as usual we assume that the residuals (ε_i) are independent and identically distributed (i.i.d.) normal random variables with zero mean and some unknown constant variance (σ^2). Note, that this is the assumption for all random variable error terms in models presented in this chapter, however it is not noted for every model.

The model in Equation (6-1) is referred to as the *General Linear Model* (GLM), and is closely related to the ANOVA covered in a later chapter. As we will see in Section 6.2, we can also use the approach to approximate non-linear functions of the regressors, i.e.

$$Y_i = f(x_i) + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2). \tag{6-2}$$

The optimal set of parameters for the multiple linear regression model is found by minimising the residual sum of squares

$$RSS(\beta_0, \ldots, \beta_p) = \sum_{i=1}^{n} \left[Y_i - (\beta_0 + \beta_1 x_{1,i} + \cdots + \beta_p x_{p,i}) \right]^2, \tag{6-3}$$

where n is the number of observations. The general problem is illustrated in Figure 6.1, where the black dots represent the observations (y_i), the blue and red lines represent errors (e_i) (the ones we minimize), and the surface represented by the grey lines is the optimal estimate (with $p = 2$)

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{1,i} + \hat{\beta}_2 x_{2,i}, \tag{6-4}$$

or

$$y_i = \hat{y}_i + e_i, \tag{6-5}$$

again we put a "hat" on the parameters to emphasize that we are dealing with parameter estimates (or estimators), as a result of minimising Equation (6-3) with respect to $\beta_0, \ldots, \beta_p$.

Let's have a look at a small example:

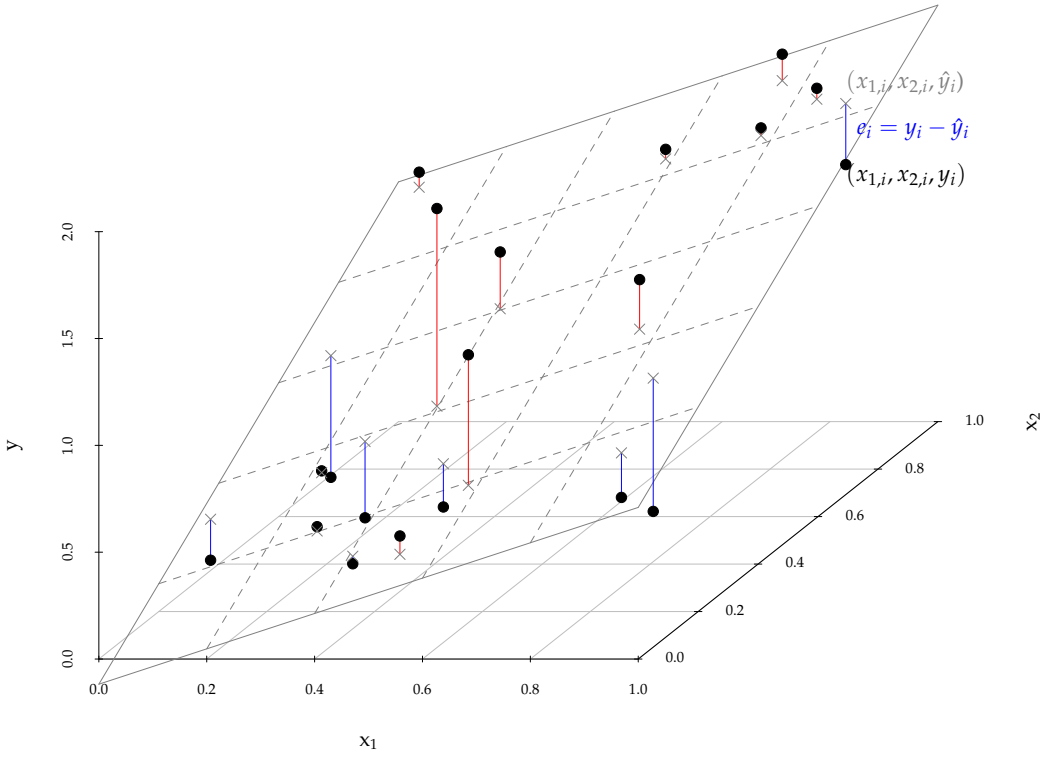


Figure 6.1: Conceptual plot for the multiple linear regression problem (red lines, $e_i > 0$, blue lines ($e_i < 0$).

Example 6.1

The car manufacture in Example 5.1 in Chapter 5 constructed a linear model for fuel consumption as a function of speed, now a residual analysis revealed that the residuals were not independent of the fitted values and therefore the model should be extended. It is realized that the fuel consumption is a function of wind speed as well as the speed of the car, and a new model could be formulated as

$$Y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \varepsilon_i \tag{6-6}$$

where $x_{1,i}$ is the speed, and $x_{2,i}$ is the wind speed (relative to the car). Another possibility is that the model should in fact not be linear in the speed, but rather quadratic

$$\begin{aligned} Y_i &= \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{1,i}^2 + \varepsilon_i && (6\text{-}7)\\ &= \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \varepsilon_i, && (6\text{-}8) \end{aligned}$$

where $x_{2,i}$ is now the squared speed. Both models ((6-6) and (6-7)) are linear in the parameters $(\beta_0, \beta_1, \beta_2)$.

The example above illustrate that linearity refers to linearity in the parameters, not the regressors. E.g. the model

$$Y_i = \beta_0 + \beta_2 \log(x_i) + \varepsilon_i, \tag{6-9}$$

is a linear model, while

$$Y_i = \beta_0 + \log(x_i + \beta_2) + \varepsilon_i, \tag{6-10}$$

is not a linear model.

6.1 Parameter estimation

Just as in the case of simple linear regression the optimal parameters are the parameters that minimize the residual sum of squares (RSS), this is equivalent to equating the partial derivatives of RSS (Equation (6-3)) with zero, i.e.

$$\frac{\partial RSS}{\partial \beta_j} = 0; \quad j = 0, 1, \ldots, p, \tag{6-11}$$

which will give us $p+1$ equations (the partial derivatives) in $p+1$ unknowns (the parameters)

$$2\sum_{i=1}^{n}\left[y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_{1,i} + \cdots + \hat{\beta}_p x_{p,i})\right] = 0, \tag{6-12}$$

$$2\sum_{i=1}^{n}\left[y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_{1,i} + \cdots + \hat{\beta}_p x_{p,i})x_{1,i}\right] = 0, \tag{6-13}$$

$$\vdots$$

$$2\sum_{i=1}^{n}\left[y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_{1,i} + \cdots + \hat{\beta}_p x_{p,i})x_{p,i}\right] = 0, \tag{6-14}$$

the Equations (6-12)-(6-14) are referred to as the normal equations, and as we can see these are a system of linear equations and thus best solved by methods of linear algebra. The matrix formulation is covered in Section 6.6, but for now we will just assume that `Python` is able to solve the normal equations and give the correct parameter estimates, standard errors for the parameter estimates, etc.

When the ε_i's are independent identically normally distributed, we can construct tests for the individual parameters, assuming we know the parameter estimates and their standard errors:

|||| Theorem 6.2 Hypothesis tests and confidence intervals

Suppose the we are given parameter estimates $(\hat{\beta}_0, \ldots, \hat{\beta}_p)$ and their corresponding standard errors $(\hat{\sigma}_{\beta_0}, \ldots, \hat{\sigma}_{\beta_p})$, then under the null hypothesis

$$H_{0,i}: \quad \beta_i = \beta_{0,i}, \tag{6-15}$$

the t-statistic

$$T_i = \frac{\hat{\beta}_i - \beta_{0,i}}{\hat{\sigma}_{\beta_i}}, \tag{6-16}$$

will follow the t-distribution with $n-(p+1)$ degrees of freedom, and hypothesis testing and confidence intervals should be based on this distribution. Further, a central estimate for the residual variance is

$$\hat{\sigma}^2 = \frac{RSS(\hat{\beta}_0, \ldots, \hat{\beta}_p)}{n-(p+1)}. \tag{6-17}$$

The interpretation of multiple linear regression in `Python` is illustrated in the

following example:

|||| Example 6.3

The data used for Figure 6.1 is given in the table below

x_1	0.083	0.409	0.515	0.397	0.223	0.292	0.584	0.491	0.923	0.280
x_2	0.625	0.604	0.077	0.414	0.343	0.202	0.840	0.266	0.831	0.385
y	0.156	1.234	0.490	1.649	0.500	0.395	1.452	0.416	1.390	0.234
x_1	0.772	0.857	0.758	0.850	0.409	0.055	0.578	0.745	0.886	0.031
x_2	0.821	0.308	0.440	0.865	0.111	0.970	0.192	0.939	0.149	0.318
y	1.574	0.349	1.287	1.709	0.323	1.201	1.210	1.787	0.591	0.110

We assume the model

$$Y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2). \tag{6-18}$$

In order to estimate parameters we would write:

```
# Read data
data = {
'x1' : [0.083, 0.409, 0.515, 0.397, 0.223, 0.292, 0.584, 0.491, 0.923,
        0.280, 0.772, 0.857, 0.758, 0.850, 0.409, 0.055, 0.578, 0.745,
        0.886, 0.031],
'x2' : [0.625, 0.604, 0.077, 0.414, 0.343, 0.202, 0.840, 0.266, 0.831,
        0.385, 0.821, 0.308, 0.440, 0.865, 0.111, 0.970, 0.192, 0.939,
        0.149, 0.318],
'y'  : [0.156, 1.234, 0.490, 1.649, 0.500, 0.395, 1.452, 0.416, 1.390,
        0.234, 1.574, 0.349, 1.287, 1.709, 0.323, 1.201, 1.210, 1.787,
        0.591, 0.110]
}

df = pd.DataFrame(data)
```

```
# Parameter estimation
fit = smf.ols(formula = 'y ~ x1 + x2', data = df).fit()

# Summary of fit (parameter estimates, standard error, p-values, etc.)
print(fit.summary(slim=True))

                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.632
Model:                            OLS   Adj. R-squared:                  0.589
No. Observations:                  20   F-statistic:                     14.62
Covariance Type:            nonrobust   Prob (F-statistic):           0.000203
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.1176      0.212     -0.556      0.586      -0.564       0.329
x1             0.8274      0.304      2.719      0.015       0.185       1.470
x2             1.2393      0.293      4.236      0.001       0.622       1.857
==============================================================================
```

The interpretation of the output is exactly the same as in the simple linear regression. The first column gives the parameter estimates $(\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2)$, second column gives the standard error of the parameter estimates $(\hat{\sigma}_{\beta_0}, \hat{\sigma}_{\beta_1}, \hat{\sigma}_{\beta_2})$, third column gives the t-statistics for the standard hypothesis $H_{0,i} : \beta_i = 0$, column four gives the p-value for the two-sided alternative, and finally columns 5-6 give 95% confidence intervals. We can therefore conclude that the effect of x_1 and x_2 are both significant on a 5% confidence level.

Method 6.4 Level α t-tests for parameters

1. Formulate the *null hypothesis*: $H_{0,i} : \beta_i = \beta_{0,i}$, and the alternative hypothesis $H_{1,i} : \beta_i \neq \beta_{0,i}$
2. Compute the test statistic $t_{\text{obs},\beta_i} = \frac{\hat{\beta}_i - \beta_{0,i}}{\hat{\sigma}_{\beta_i}}$
3. Compute the evidence against the *null hypothesis*

$$p\text{-value}_i = 2P(T > |t_{\text{obs},\beta_i}|) \tag{6-19}$$

4. If the p-value$_i < \alpha$ reject $H_{0,i}$, otherwise accept $H_{0,i}$

In many situations we will be more interested in quantifying the uncertainty of the parameter estimates rather than testing a specific hypothesis. This is usually given in the form of confidence intervals for the parameters:

Method 6.5 Parameter confidence intervals

$(1 - \alpha)$ confidence interval for β_i is given by

$$\hat{\beta}_i \pm t_{1-\alpha/2}\, \hat{\sigma}_{\beta_i}, \tag{6-20}$$

where $t_{1-\alpha/2}$ is the $(1 - \alpha/2)$-quantile of a t-distribution with $n - (p + 1)$ degrees of freedom.

Remark 6.6 (On finding $\hat{\beta}_i$ and $\sigma_{\hat{\beta}_i}$ in methods 6.4 and 6.5)

In Chapter 5 we were able to formulate the exact formulas for $\hat{\beta}_i$ and $\hat{\sigma}_{\hat{\beta}_i}$, in a multiple linear regression setting we simply use `Python` (`smf.ols`), to find these values.
The explicit formulas are however given in the matrix formulation of the linear regression problem in Section 6.6.

Example 6.7

For our example the 95% confidence intervals become ($t_{1-\alpha/2} = 2.110$)

$$I_{\beta_0} = -0.118 \pm 2.110 \cdot 0.212, \tag{6-21}$$
$$I_{\beta_1} = 0.827 \pm 2.110 \cdot 0.304, \tag{6-22}$$
$$I_{\beta_2} = 1.239 \pm 2.110 \cdot 0.293, \tag{6-23}$$

or using the software (for β_0):

```
# Calculations
CI = - 0.118 + np.array([-1,1]) * stats.t.ppf(0.975, df = 17) * 0.212
print(CI)

[-0.565  0.329]
```

or directly using the highlevel method (for β_0, β_1, and β_2):

```
print(fit.conf_int(alpha=0.05))

                   0         1
Intercept -0.564307  0.329042
x1         0.185371  1.469529
x2         0.621989  1.856559
```

The examples above illustrates how we can construct confidence intervals for the parameters and test hypotheses without having to implement the actual estimation ourselves.

6.1.1 Confidence and prediction intervals for the line

Just as for the simple linear regression model we will often be interested in prediction of future outcome of an experiment, and as usual we will be interested in quantifying the uncertainty of such an experiment. The expected value of a new experiment (with $x_1 = x_{1,\text{new}}, \ldots, x_p = x_{p,\text{new}}$) is

$$\hat{y}_{\text{new}} = \hat{\beta}_0 + \hat{\beta}_1 x_{1,\text{new}} + \ldots + \hat{\beta}_p x_{p,\text{new}}. \tag{6-24}$$

In order to quantify the uncertainty of this estimate we need to calculate the

variance of $\hat{y}_{\text{new}}$, in Section 5.3 we saw that this variance is a function of: 1) the variance of the parameters, 2) the covariance between the parameters, and 3) x_{new}. This is also true in the multiple linear regression case, except that x_{new} is now a vector and we need to account for pairwise covariance between all parameter estimators. This analysis is most elegantly done with matrix formulation and is covered in Section 6.6. We can however do this using `Python` without dealing with the covariances explicitly.

This is illustrated in the following example:

|||| Example 6.8

With reference to Example 6.3 suppose we want to predict the expected value of Y at $(x_{1,\text{new}}, x_{2,\text{new}}) = (0.5, 0.5)$ and at $(x_{1,\text{new}}, x_{2,\text{new}}) = (1,1)$, we would also like to know the standard error of the prediction and further the confidence and the prediction intervals. The standard error of the prediction can be calculated by:

```
# # New data
new_data = pd.DataFrame({'x1': [0.5, 1],'x2': [0.5, 1]})

# # Prediction and confidence interval
pred = fit.get_prediction(new_data).summary_frame(alpha=0.05)
print(round(pred,3))
```

```
    mean  mean_se  mean_ci_lower  mean_ci_upper  obs_ci_lower  obs_ci_upper
0  0.916    0.085          0.737          1.095         0.098         1.734
1  1.949    0.214          1.497          2.401         1.032         2.867
```

The data-frame "`new_data`" is the points where we want to predict the outcome, the object "`pred`" has the fitted values (`mean`) at the points in "`new_data`", the standard errors for the predictions (`mean_se`), the upper and lower limits of the confidence intervals (`mean_ci_upper` and `mean_ci_lower`), and the upper and lower limits of the prediction intervals (`obs_ci_upper` and `obs_ci_lower`).

Notice that the standard error for $\hat{y}_{\text{new}}$ is much larger for the point $(x_{1,\text{new}}, x_{2,\text{new}}) = (1,1)$ than for the point $(x_{1,\text{new}}, x_{2,\text{new}}) = (0.5, 0.5)$, this is because the (1,1) point is far from the average of the regressors, while the point (0.5,0.5) is close to the average value of the regressors.

Now, we are actually able to calculate confidence and prediction intervals for the two points, the confidence intervals become

$$CI_1 = 0.9157 \pm t_{1-\alpha/2} \cdot 0.08477, \tag{6-25}$$

$$CI_2 = 1.9491 \pm t_{1-\alpha/2} \cdot 0.21426, \tag{6-26}$$

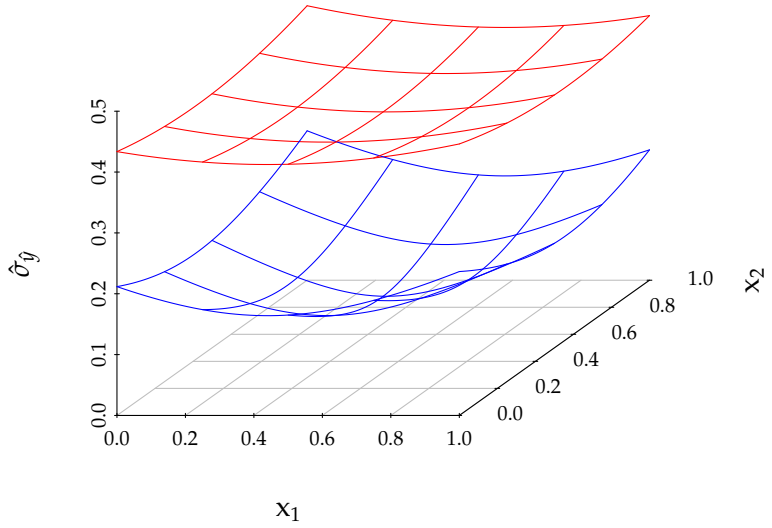


Figure 6.2: Standard error for $\hat{y}_{\text{new}}$ (blue surface) and standard error for y_{new} (red surface).

and the prediction intervals become (add the variance of $\hat{Y}_{\text{new}}$ and $\hat{\sigma}^2$)

$$PI_1 = 0.9157 \pm t_{1-\alpha/2} \cdot \sqrt{0.08477^2 + 0.3784^2}, \tag{6-27}$$

$$PI_2 = 1.9491 \pm t_{1-\alpha/2} \cdot \sqrt{0.21426^2 + 0.3784^2}, \tag{6-28}$$

where $t_{1-\alpha/2}$ is obtained from a t-distribution with 17 degrees of freedom.

The calculations in `Python` is exemplified for the first prediction interval below

```
p_i = 0.9157 + np.array([-1,1]) * stats.t.ppf(0.975, df=17) *
np.sqrt(0.08477**2 + 0.3784**2)
np.round(p_i, 3)

array([0.098, 1.734])
```

We saw in the example above that the standard error for the fit is large when we are far from the center of mass for the regressors, this is illustrated in Figure 6.2.

Method 6.9 Intervals for the line (by Python)

The (1-α) **confidence and prediction intervals** for the line $\hat{\beta}_0 + \hat{\beta}_1 x_{1,\text{new}} + \cdots + \hat{\beta}_p x_{p,\text{new}}$ are calculated in `Python` by

```
# Confidence and Prediction interval
fit.get_prediction(new_data).summary_frame(alpha=alpha)
```

Remark 6.10

Explicit formulas for confidence and prediction intervals are given in Section 6.6.

6.2 Curvilinear regression

Suppose we are given pairs of values of x and y and there seems to be information in x about y, but the relation is clearly non-linear

$$Y_i = f(x_i) + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2), \tag{6-29}$$

and the non-linear function $f(x)$ is unknown to us. The methods we have discussed don't apply for non-linear functions, and even if we could do non-linear regression we would not know which function to insert. We do however know from elementary calculus that any function can be approximated by its Taylor series expansion

$$f(x) \approx f(0) + f'(0) \cdot x + \frac{f''(0)}{2} x^2 + \cdots + \frac{f^{(p)}(0)}{p!} x^p, \tag{6-30}$$

now replace the Taylor series coefficients $\left(\frac{f^{(j)}(0)}{j!}\right)$ by β_j and insert (6-30) in (6-29) to get

$$\begin{aligned} Y_i &= \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x^p + \varepsilon_i \\ &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon_i, \end{aligned} \tag{6-31}$$

where $x_j = x^j$, we refer to this method as curvilinear regression. The method is illustrated in the following example:

Example 6.11 Simulation of non-linear model

We simulate the following model

$$Y_i = \sin(\pi x_i) + \varepsilon_i, \quad \varepsilon_i \sim N(0, 0.1^2), \tag{6-32}$$

with $x \in [0,1]$ by:

```
np.random.seed(12657)
n = 200
x = np.random.uniform(size = n)
y = np.sin(np.pi * x) + np.random.normal(0, 0.1,size=n)
df_sim = pd.DataFrame({'y': y,'x1' : x, 'x2' : x**2})
```

Y_i is a non-linear function of x but lets try to estimate parameters in the simple linear regression model

$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2), \tag{6-33}$$

and find the 95% confidence interval for the parameters:

```
fit_sim = smf.ols(formula = 'y ~ x1', data = df_sim).fit()
print(round(fit_sim.conf_int(alpha=0.05),3))
```

```
               0      1
Intercept  0.510  0.690
x1        -0.097  0.211
```

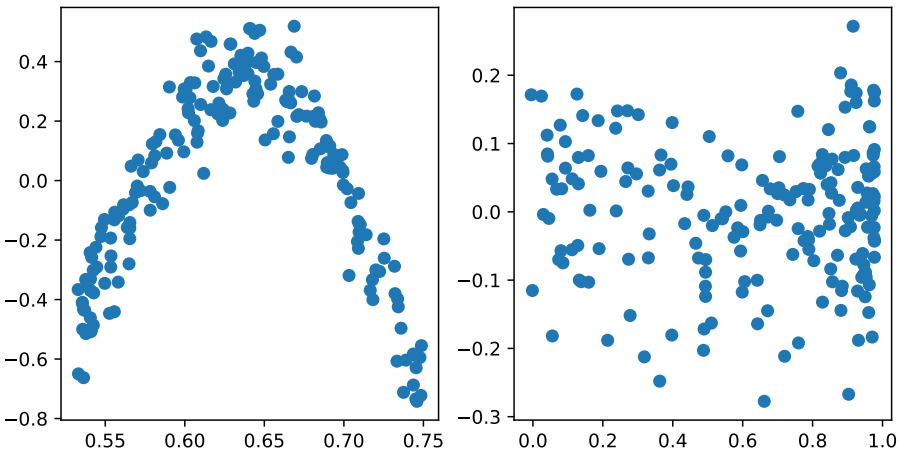
We see that the 95% confidence interval for β_1 covers zero, and we can therefore not reject the null hypothesis that β_1 is zero. Now include a quadratic term in x_1 to approximate the non-linear function by the model

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2), \tag{6-34}$$

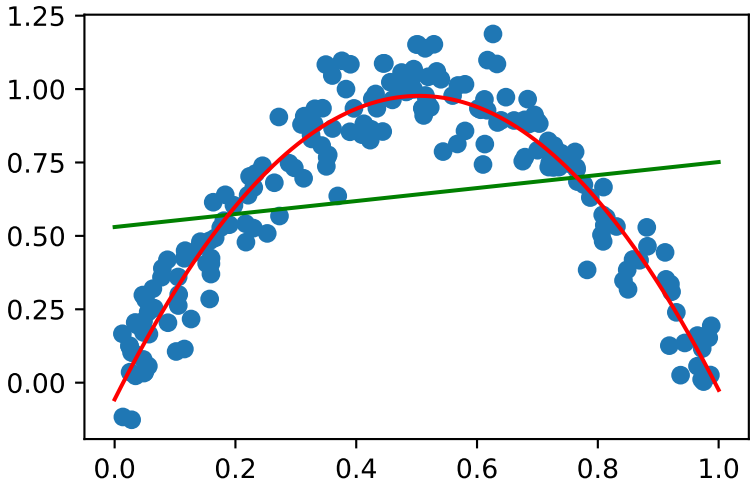
```
fit_sim2 = smf.ols(formula = 'y ~ x1 + x2', data = df_sim).fit()
print(round(fit_sim2.conf_int(alpha=0.05),3))
```

```
               0      1
Intercept -0.095 -0.006
x1         3.885  4.303
x2        -4.292 -3.883
```

Now we see that all parameters are significantly different from zero on a 5% confidence level. The plot below shows the residuals for the two models as a function of the fitted values:



It is clear that including the second order term removed most, if not all, systematic dependence in the residuals. Also looking at the fitted values together with the actual values shows that we have a much better model when including the second order term (red line):



Remark 6.12

In general one should be careful when extrapolation models into areas where there is no data, and this is in particular true when we use curvilinear regression.

6.3 Collinearity

In statistics collinearity refers to situations where the sample correlation between the independent variables is high. If this is the case we should be careful with interpretation of parameter estimates, and often we should actually reduce the model. Now consider the model

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2), \tag{6-35}$$

and assume that the sample correlation between x_1 and x_2 is exactly equal 1, this implies that we can write $x_2 = a + bx_1$, inserting in (6-35) gives

$$\begin{aligned} y_i &= \beta_0 + \beta_1 x_1 + \beta_2(a + bx_1) + \varepsilon_i & (6\text{-}36)\\ &= \beta_0 + \beta_2 a + (\beta_1 + \beta_2 b)x_1 + \varepsilon_i, & (6\text{-}37) \end{aligned}$$

which shows that we can only identify $\beta_0 + \beta_2 a$ and $(\beta_1 + \beta_2 b)$, so the model is essentially a simple linear regression model. It could also have been the other way around, i.e. $x_1 = a + bx_2$, and thus it seems that it is not possible to distinguish between x_1 and x_2. In real life application the correlation between the regressors is rarely 1, but rather close to 1 and we need to handle this case as well. In actual practice a simple way to handle this is, by adding or removing one parameter at the time. Other procedures exist, e.g. using the average of the regressors, or using principle component regression, we will not discuss these approaches further here.

A small example illustrates the principle:

Example 6.13 Simulation

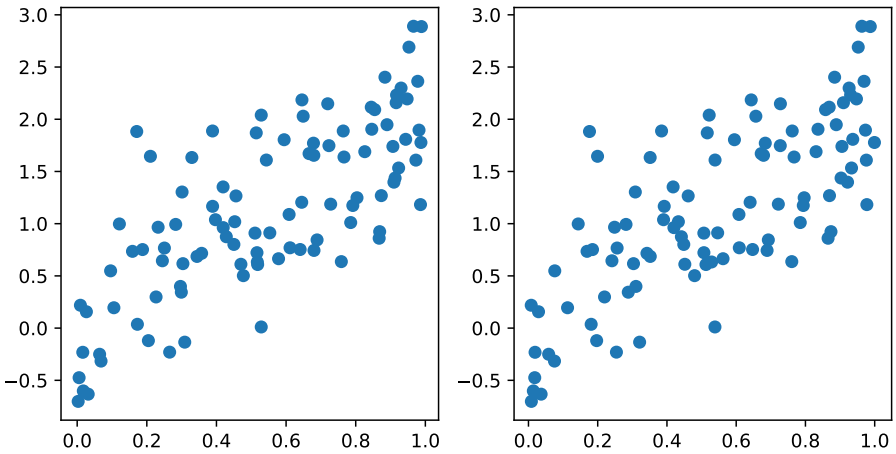
Consider the model

$$Y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2), \tag{6-38}$$

with data generated from the following code:

```
np.random.seed(200)
n = 100
x1 = np.random.uniform(size = n)
x2 = x1 + np.random.normal(0, 0.01,size=n)
y = x1 + x2 + np.random.normal(0, 0.5,size=n)
df_sim = pd.DataFrame({'y': y,'x1' : x1, 'x2' : x2})

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8,4))
ax1.scatter(x1,y)
ax2.scatter(x2,y)
```



Clearly, both x_1 and x_2 contain information about y, but our usual linear regression gives:

```
fit_sim = smf.ols(formula = 'y ~ x1 + x2', data = df_sim).fit()
print(round(fit_sim.conf_int(alpha=0.05),3))
```

```
                0       1
Intercept  -0.197   0.247
x1        -14.847  10.061
x2         -8.057  16.898
```

we see that none of the parameters are significant (on a 5% level), but if we remove x_1 (this is the one with the highest p-value) from the model we get:

```
fit_sim2 = smf.ols(formula = 'y ~ x2', data = df_sim).fit()
print(fit_sim2.summary(slim=True))

                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.567
Model:                            OLS   Adj. R-squared:                  0.562
No. Observations:                 100   F-statistic:                     128.2
Covariance Type:            nonrobust   Prob (F-statistic):           1.69e-19
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.0283      0.111      0.255      0.799      -0.192       0.249
x2             2.0240      0.179     11.322      0.000       1.669       2.379
==============================================================================
```

and the slope is now highly significant.

The lesson learned from the example above is that we should always try to reduce the model before concluding that individual parameters are zero. Model development is a partly manual process, where the end result might depend on the selection strategy. The usual strategies are: *backward selection*, where we start by the most complicated model we can think of and remove one term at a time (this is what we did in the example above), and *forward selection* where we start by a simple model and include new terms one by one.

|||| Remark 6.14 Interpretation of parameters

In general we can interpret the parameters of a multiple linear regression model as the effect of the variable given the other variables. E.g. β_j is the effect of x_j when we have accounted for other effects (x_i, $i \neq j$). This interpretation is however problematic when we have strong collinearity, because the true effects are hidden by the correlation.

An additional comment on the interpretation of parameters in the example above is: since the data is simulated, we know that the true parameters are $\beta_1 = \beta_2 = 1$. In the full model we got $\hat{\beta}_1 \approx$-2.40 and $\hat{\beta}_2 \approx 4.42$. Both of these numbers are clearly completely off, the net effect is however $\hat{\beta}_1 + \hat{\beta}_2 \approx$2.02 (because $x_1 \approx x_2$). In the reduced model we got $\hat{\beta}_2 =$2.02, which is of course also wrong, but nearly the same level, and only holds because $x_1 \approx x_2$.

6.4 Residual analysis

Just as for the simple linear regression model we will need to justify that the assumptions in the linear regression model holds. This is handled by q-q plots, and considering the relation between the residuals and the fitted values. This analysis is exactly the same as for the simple linear regression in Section 5.7.

We saw that plotting the residuals as a function of fitted values could reveal systematic dependence, which imply there are un-modelled effects that should be included in the model. The question is of course how we can identify such effects. One way is to plot the residuals as a function of potential regressors, which are not included. Plotting the residuals as a function of the included regressors might reveal non-linear effects. Again we illustrate this method by an example:

|||| Example 6.15 Residuals analysis

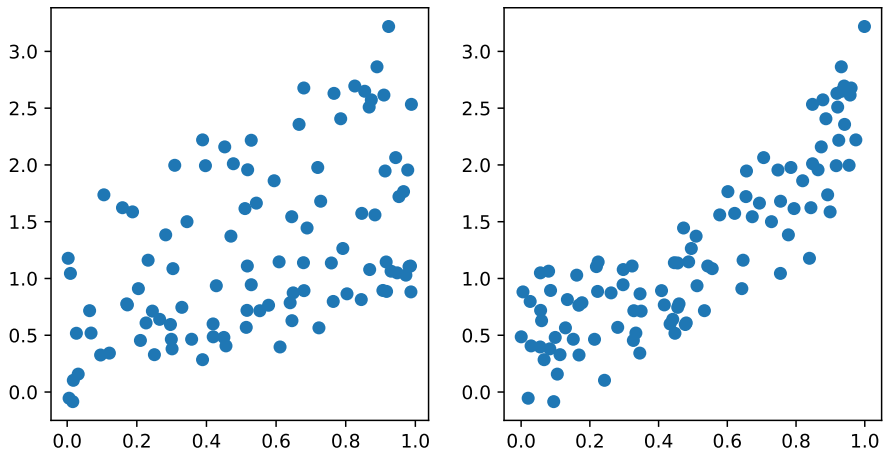
Consider the model in the Python script below, the true model is

$$y_i = x_1 + 2x_2^2 + \varepsilon_i, \quad \varepsilon_i \sim N(0, 0.125^2) \tag{6-39}$$

in a real application the true model is of course hidden to us and we would start by a multiple linear model with the two effects x_1 and x_2. Looking at the plots below also suggests that this might be a good model:

```
np.random.seed(200)
n = 100
x1 = np.random.uniform(size = n)
x2 = np.random.uniform(size = n)
y = x1 + 2*x2**2 + + np.random.normal(0, 0.125,size=n)
df_sim = pd.DataFrame({'y': y,'x1' : x1, 'x2' : x2})

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8,4))
ax1.scatter(x1,y)
ax2.scatter(x2,y)
```



Now we fit the model

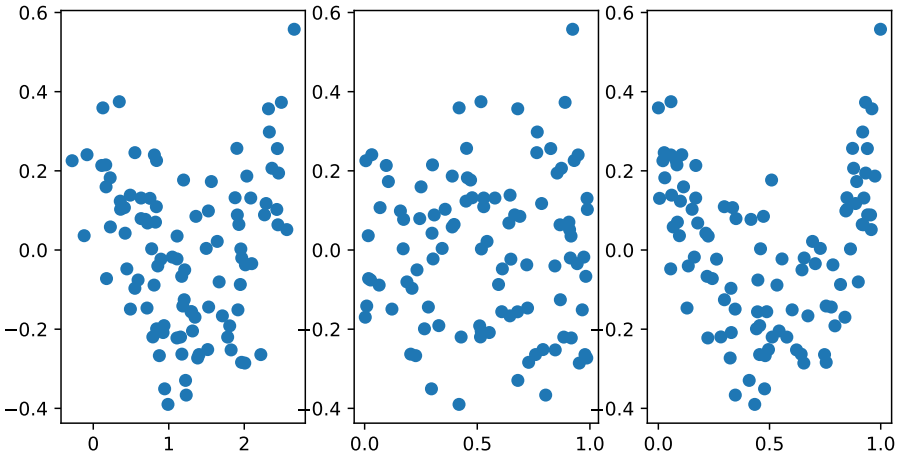
$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2), \tag{6-40}$$

and plot the resulting residuals as a function of the fitted values, and the independent variables (x_1 and x_2). There seems to be a systematic dependence between the fitted values and the residuals (left plot):

```
fit_sim = smf.ols(formula = 'y ~ x1 + x2', data = df_sim).fit()

res = y - fit_sim.fittedvalues
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(8,4))

ax1.scatter(fit_sim.fittedvalues,res)
ax2.scatter(x1,res)
ax3.scatter(x2,res)
```

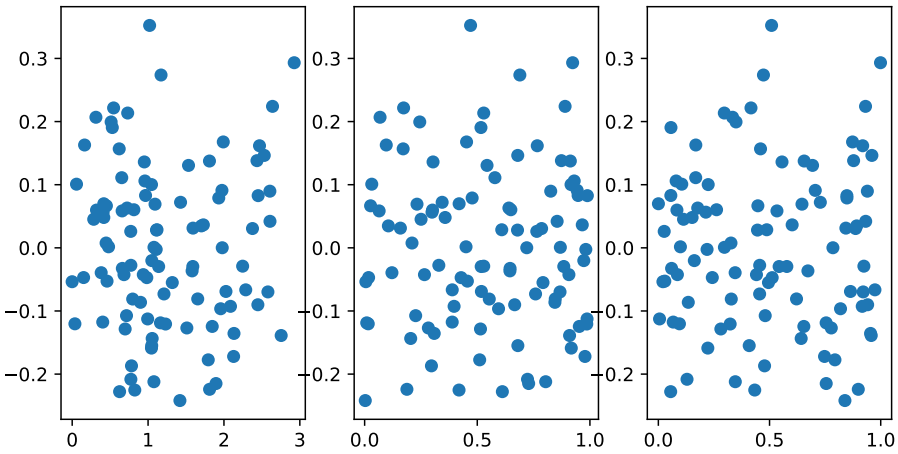


The left plot does however not suggest where the dependence comes from. Now looking at the residuals as a function of x_1 and x_2 (centre and left plot) reveal that the residuals seem to be quadratic in x_2, and we should therefore include x_2^2 in the model:

```
x3 = x2**2
df_sim = pd.DataFrame({'y': y,'x1' : x1, 'x2' : x2,'x3': x3})
fit_sim = smf.ols(formula = 'y ~ x1 + x2 +x3', data = df_sim).fit()

res = y - fit_sim.fittedvalues

fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(8,4))
ax1.scatter(fit_sim.fittedvalues,res)
ax2.scatter(x1,res)
ax3.scatter(x2,res)
```



We now see that there is no systematic dependence in the residuals and we can report the final result.

```
print(fit_sim.summary(slim=True))
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.971
Model:                            OLS   Adj. R-squared:                  0.970
No. Observations:                 100   F-statistic:                     1057.
Covariance Type:            nonrobust   Prob (F-statistic):           2.33e-73
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.0100      0.044     -0.230      0.819      -0.097       0.077
x1             1.0166      0.044     23.150      0.000       0.929       1.104
x2             0.1342      0.173      0.774      0.441      -0.210       0.478
x3             1.8668      0.169     11.056      0.000       1.532       2.202
==============================================================================
```

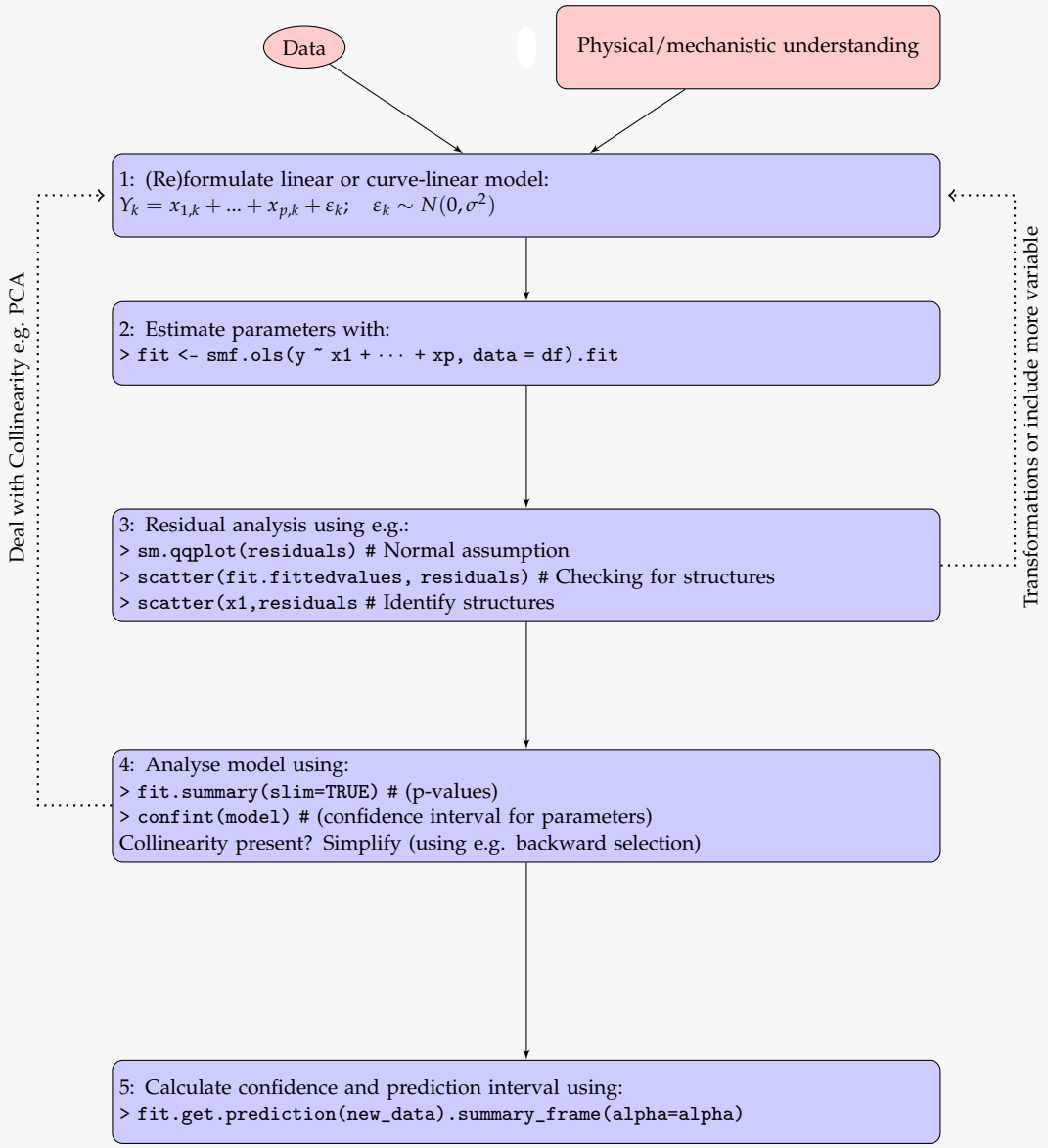
Now we can actually see that we find parameter values close to the true ones, further the slope related to x_2 and the intercept is not significant, however usually when x_2^2 have a significant parameter we would also keep x_2 in the model, the same comment apply to the intercept, that we would usually always include in the model.

6.5 Linear regression in Python

Method 6.16 below gives a practical summary of Chapter 5 and 6 with references to the applied R-functions.

Method 6.16

This method box is a very short guide to `Python` and linear regression.



6.6 Matrix formulation

The multiple linear regression problem can be formulated in vector-matrix notation as

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim N(\boldsymbol{0}, \sigma^2 \boldsymbol{I}), \tag{6-41}$$

or

$$\begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{p,1} \\ \vdots & \vdots & & \vdots \\ 1 & x_{1,n} & \cdots & x_{p,n} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_p \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{bmatrix}, \qquad \varepsilon_i \sim N(0, \sigma^2). \tag{6-42}$$

Notice, that the formulation in (6-41) is exactly the same as we saw in Section 5.5.

The residual sum of squares are calculated by

$$RSS = \boldsymbol{\varepsilon}^T \boldsymbol{\varepsilon} = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}), \tag{6-43}$$

and the parameter estimates are given by:

|||| Theorem 6.17

The estimators of the parameters in the simple linear regression model are given by

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{Y}, \tag{6-44}$$

and the covariance matrix of the estimates is

$$V[\hat{\boldsymbol{\beta}}] = \sigma^2 (\boldsymbol{X}^T \boldsymbol{X})^{-1}, \tag{6-45}$$

and central estimate for the residual variance is

$$\hat{\sigma}^2 = \frac{RSS}{n - (p+1)}. \tag{6-46}$$

The proof of this theorem follows the exact same arguments as the matrix formulation of the simple linear regression model in Chapter 5 and hence it is omitted here.

Marginal tests ($H_0 : \beta_i = \beta_{i,0}$) can also in the multiple linear regression case be

constructed by

$$\frac{\hat{\beta}_i - \beta_{i,0}}{\sqrt{(\hat{\mathbf{\Sigma}}_\beta)_{ii}}} \sim t(n-(p+1)). \tag{6-47}$$

6.6.1 Confidence and prediction intervals for the line

Now suppose that we want to make a prediction at a new point

$$\boldsymbol{x}_{\text{new}} = [1, x_{1,\text{new}}, \ldots, x_{p,\text{new}}],$$

in order to construct confidence and prediction intervals we calculate the variance of $\hat{Y}_{\text{new}}$

$$\begin{aligned} \mathrm{V}(\hat{Y}_{\text{new}}) &= \mathrm{V}(\boldsymbol{x}_{\text{new}}\hat{\boldsymbol{\beta}}) \\ &= \boldsymbol{x}_{\text{new}}\,\mathrm{V}(\hat{\boldsymbol{\beta}})\boldsymbol{x}_{\text{new}}^T \\ &= \sigma^2\boldsymbol{x}_{\text{new}}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{x}_{\text{new}}^T, \end{aligned} \tag{6-48}$$

in practice we will of course replace σ^2 with its estimate ($\hat{\sigma}^2$), and hence use quantile of the appropriate t-distribution (and standard errors rather than variances) to calculate confidence intervals. The variance of a single prediction is calculated by

$$\begin{aligned} \mathrm{V}(Y_{\text{new}}) &= \mathrm{V}(\boldsymbol{x}_{\text{new}}\hat{\boldsymbol{\beta}} + \varepsilon_{\text{new}}) \\ &= \boldsymbol{x}_{\text{new}}\,\mathrm{V}(\hat{\boldsymbol{\beta}})\boldsymbol{x}_{\text{new}}^T + \sigma^2 \\ &= \sigma^2(1 + \boldsymbol{x}_{\text{new}}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{x}_{\text{new}}^T). \end{aligned} \tag{6-49}$$

The calculations above illustrate that the derivations of variances are relatively simple, when we formulate our model in the matrix-vector notation.

Glossaries

Alternative hypothesis [Alternativ hypotese] The alternative hypothesis (H_1) is ofter the negation of the null hypothesis 7

Correlation [Korrelation] The sample correlation coefficient are a summary statistic that can be calculated for two (related) sets of observations. It quantifies the (linear) strength of the relation between the two. See also: Covariance 14, 16

Covariance [Kovarians] The sample covariance coefficient are a summary statistic that can be calculated for two (related) sets of observations. It quantifies the (linear) strength of the relation between the two. See also: Correlation 9, 22

Degrees of freedom [Frihedsgrader] The number of "observations" in the data that are free to vary when estimating statistical parameters often defined as $n - 1$ 4, 7, 10

Linear regression [Lineær regression (-sanalyse)] 3, 6, 8, 12, 15, 17, 22

Multiple linear regression [Multipel lineær regression (-sanalyse)] 1, 4, 7, 9, 22

Null hypothesis [Nulhypotese (H_0)] 4, 7, 12

***P*-value** [*p*-værdi (for faktisk udfald af en teststørrelse)] 6, 7, 15

Acronyms

ANOVA Analysis of Variance *Glossary:* Analysis of Variance

cdf cumulated distribution function *Glossary:* cumulated distribution function

CI confidence interval 4, 7–9, 12, 23, *Glossary:* confidence interval

CLT Central Limit Theorem *Glossary:* Central Limit Theorem

IQR Inter Quartile Range *Glossary:* Inter Quartile Range

LSD Least Significant Difference *Glossary:* Least Significant Difference

pdf probability density function *Glossary:* probability density function