Universität Zürich[UZH]

Soziologisches Institut

# Data Analysis – Advanced Statistics with Python

Dr. Julia Jerke

jerke@soziologie.uzh.ch

Thursday, 12.15pm – 13.45pm, AND 2.46

# Session 3 – Linear regression

## Agenda

1. Linear regression basics
2. Linear regression with *Python*
3. Hands on

# 1. Linear regression basics

# Motivation

- In many situations we are interested in the relationship between two or more variables
- Correlational analysis can help to determine the strength of these relationships
- If we have an idea about the direction of the dependency between the variables, regression methods can be applied
- The two main objectives of regression analyses:
    1. **Description**: what is the relationship between two or more variables
        - ➢ E.g., can we explain the relationship between someone's income and someone's education, social origin, gender, etc.?
    2. **Prediction**: predicting an outcome based on various variables
        - ➢ E.g., can we predict the income of someone if we know their education, social origin, gender, etc.?

# Classic linear model

- Specifying a linear relationship between a continuous dependent variable $Y$ and one or more explanatory variables $X_1, X_2, X_3, \dots$
  - *Simple linear regression:* one explanatory variable $X_1$
  - *Multiple linear regression:* two or more explanatory variables $X_1, X_2, \dots, X_p$:
- The regression model is then given as:

$$y_i = \beta_0 + \beta_1 * x_{i1} + \beta_2 * x_{i2} + \cdots + \beta_p * x_{ip} + \epsilon_i, \qquad i = 1, \dots, n$$

whereas:

  - $\beta_0$ is the constant (i.e., the predicted value if $x_{i1}, \dots, x_{ip} = 0$)
  - $\beta_1, \dots, \beta_p$ are the regression coefficients (i.e., the effect of a one unit change in $X_1, X_2, \dots, X_p$, ceteris paribus)
  - $\epsilon_i$ is the individual error term (i.e., the difference between the true value $y_i$ and the predicted value $\hat{y}_i$)
- The coefficients $\beta_0, \beta_1, \dots, \beta_p$ are estimated in such way that they fit the data best and produce the smallest total error

# Two key questions

1. What is the best regression model to fit the data?

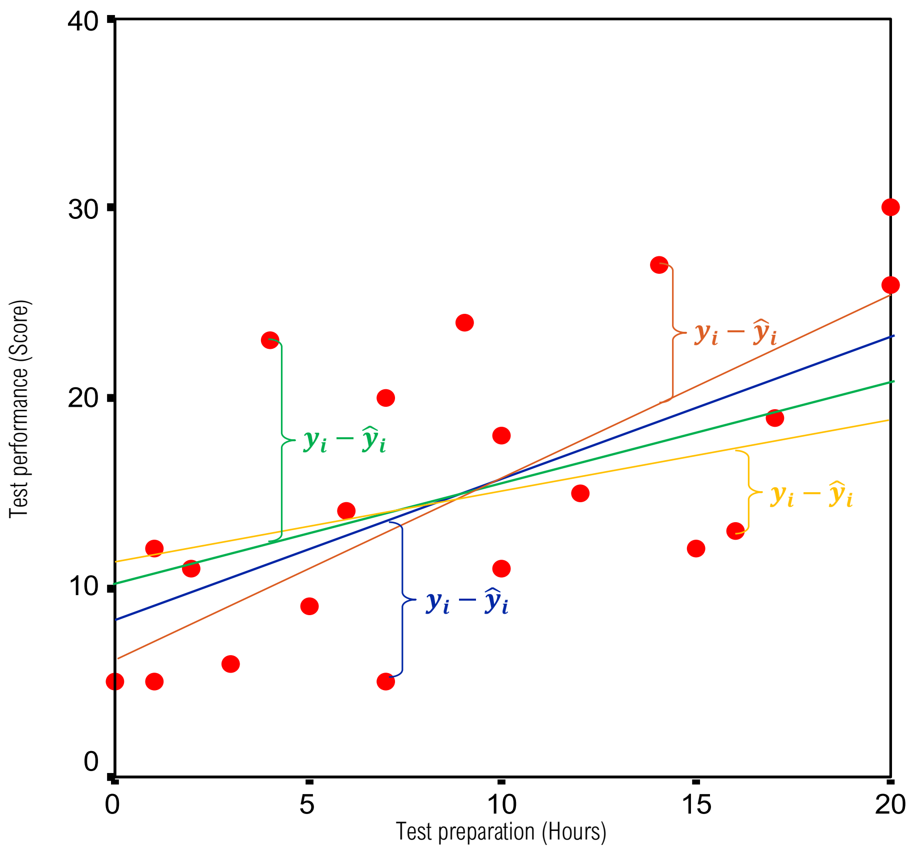
2. How good is the best regression model?

# What is the best regression model to fit the data?

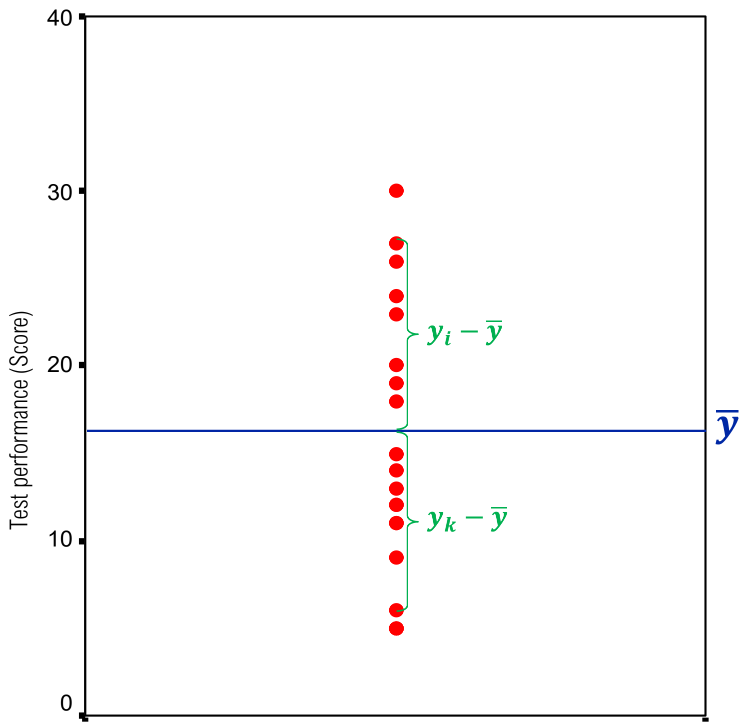
**But… what means best?**

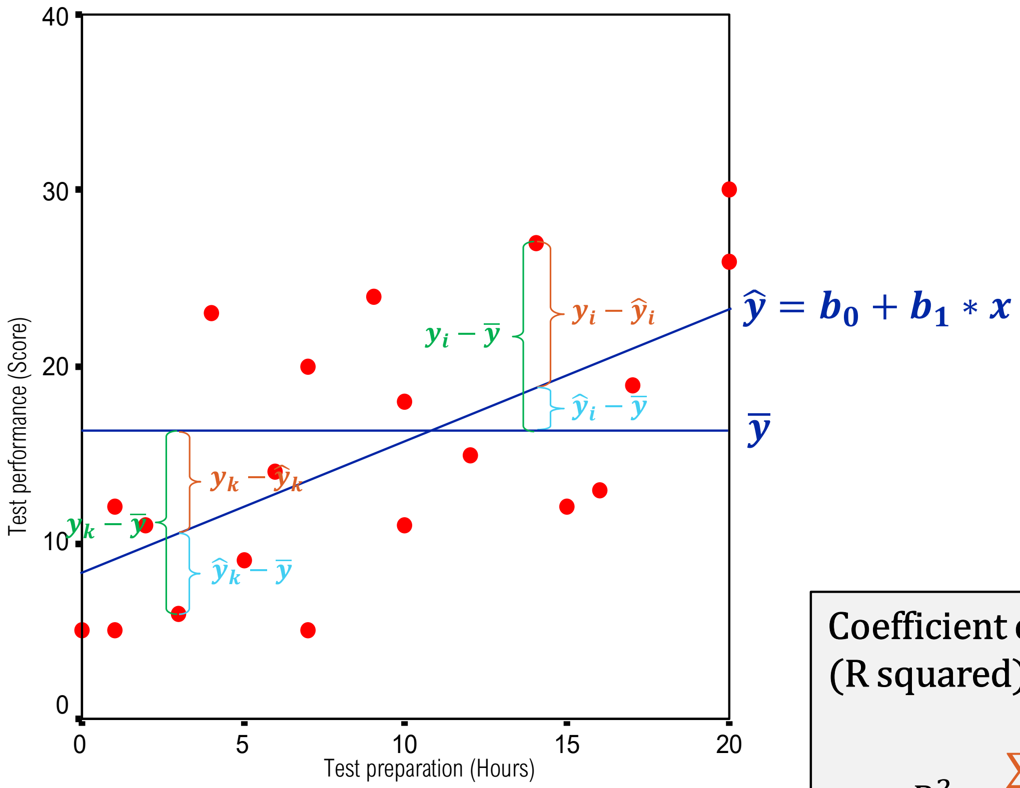
- Convention: ordinary least squares (OLS)
- The line that minimizes the *Total Squared Error* for all data points
- We, therefore, have an optimization problem:

$$\sum_{i=1}^{n} {\epsilon_i}^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \longrightarrow \min$$



# How good is the best regression model?




Coefficient of determination (R squared):

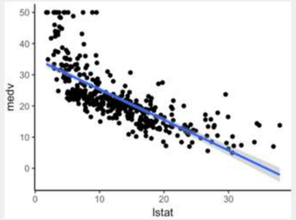
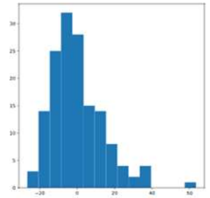
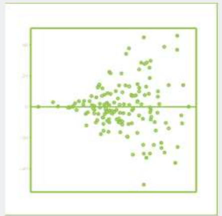
$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Variance of $Y$: $\sum_{i=1}^{n}(y_i - \bar{y})^2$

Variance that is explained by the regression model: $\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2$

Variance that remains unexplained: $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

# Requirements for the linear regression

| **Requirement** | **How does it look like if the requirement is not met?** |
|---|---|
| *Linearity*:<br>The relationship between $Y$ and $X_1, X_2, \ldots, X_p$ can be described with a linear function |  |
| *Normal distribution:*<br>The error terms follows a normal distribution with mean 0 |  |
| *Homoscedasticity*:<br>The error terms have a constant variance |  |
| *Independence*:<br>The error terms do not correlate with each other or with the variables in the model | |

# Linear regression with dummy variables

- In the classic case, the dependent and independent variables are continuous
- However, the independent variables can also be dummy or categorical variables
  - ➢ In the example of the test performance and the preparation time we could, for instance, also include the effect of the gender (*)
- Assume we have a binary variable $D$
- The interpretation of the coefficient for the dummy variable can be expressed as follows:

| $\hat{y} = b_0 + b_1 * x + b_2 * d$ | $d = 0$ | $d = 1$ |
|---|---|---|
| $\hat{y} =$ | $b_0 + b_1 * x$ | $(b_0 + b_2) + b_1 * x$ |

➢ $b_0$ is the expected difference in $Y$ between the two values of $D$ (ceteris paribus)

(*) This serves just as a simplistic example. Today, gender is rightfully no longer asked in a binary way in most surveys.

# Linear regression with interaction variables

- What if the relationship between $Y$ and $X$ differs in different groups
  - In the example of the test performance and the preparation time we could, for instance, also analyse whether the effect of the preparation time is different for female and male students
- We can include an interaction term $\mathrm{X} * D$ into the regression model
- The interpretation of the coefficient for the dummy variable can be expressed as follows:

| $\hat{y} = b_0 + b_1 * x + b_2 * d + b_3 * x \cdot d$ | $d = 0$ | $d = 1$ |
|---|---|---|
| $\hat{y} =$ | $b_0 + b_1 * x$ | $(b_0 + b_2) + (b_1 + b_3) * x$ |

  - $b_3$ is the expected difference in the effect of $\mathrm{X}$ on $Y$ between the two values of $D$ (ceteris paribus)
  - $b_2$ is the expected difference in the intercept between the two values of $D$

# 2. Linear regression in Python

## *Libraries*

- There are various ways to run a regression in Python
- Several libraries have implemented regression modelling

```
import statsmodels.formula.api as smf
```

- For instance, one such library is *statsmodel*:
  - Linear models
  - Non-linear models
  - Time series analysis
  - Event history analysis
  - Prediction and diagnostic
- Major advantage: enables the formula-style of R (`income ~ education + age + gender`)
- Extensive documentation for the library: https://www.statsmodels.org/stable/index.html

Even in statsmodel there are various modules and ways to run a regression. One way is by importing the following module:

```
import statsmodels.formula.api as smf
```

statsmodels

# Running a regression in *statsmodel*

1. Use the model class to describe the model
2. Fit the model using a class method
3. Inspect the results using a summary method

```
In [151]: model = smf.ols("score ~ gdp", data = happy) # Describing the model

In [152]: results = model.fit() # Fitting the model

In [153]: print(results.summary()) # Summarizing the model
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  score   R-squared:                       0.630
Model:                            OLS   Adj. R-squared:                  0.628
Method:                 Least Squares   F-statistic:                     262.5
Date:                Thu, 07 Oct 2021   Prob (F-statistic):           4.32e-35
Time:                        10:45:53   Log-Likelihood:                -159.97
No. Observations:                 156   AIC:                             323.9
Df Residuals:                     154   BIC:                             330.0
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      3.3993      0.135     25.120      0.000       3.132       3.667
gdp            2.2181      0.137     16.202      0.000       1.948       2.489
==============================================================================
Omnibus:                        1.139   Durbin-Watson:                   1.378
Prob(Omnibus):                  0.566   Jarque-Bera (JB):                1.244
Skew:                          -0.177   Prob(JB):                        0.537
Kurtosis:                       2.742   Cond. No.                         4.77
==============================================================================
```

# Predictions and confidence intervals

- With `results = model.fit()` we create an object that contains several attributes such as the estimated parameters, the r square, but also the predicted values and the error terms (hint: type `dir(results)` to see a full list of all attributes)

```
In [169]: predict = results.fittedvalues # accessing the predicted values y_hat

In [170]: residual = results.resid # accessing the error terms
```

- Accessing confidence and prediction intervals is a bit more complicated but also easily possible

```
# Call a more comprehensive prediction summary
prediction = results.get_prediction()

# Confidencde interval
# Access the lower value of the CI
ci_lower = prediction.summary_frame()["mean_ci_lower"]
# Access the upper value of the CI
ci_upper = prediction.summary_frame()["mean_ci_upper"]


# Prediction interval
# Access the lower value of the CI
pred_lower = prediction.summary_frame()["obs_ci_lower"]
# Access the upper value of the CI
pred_upper = prediction.summary_frame()["obs_ci_upper"]
```

# 3. Hands on

**… Open *Session_3_linear_regression.py***