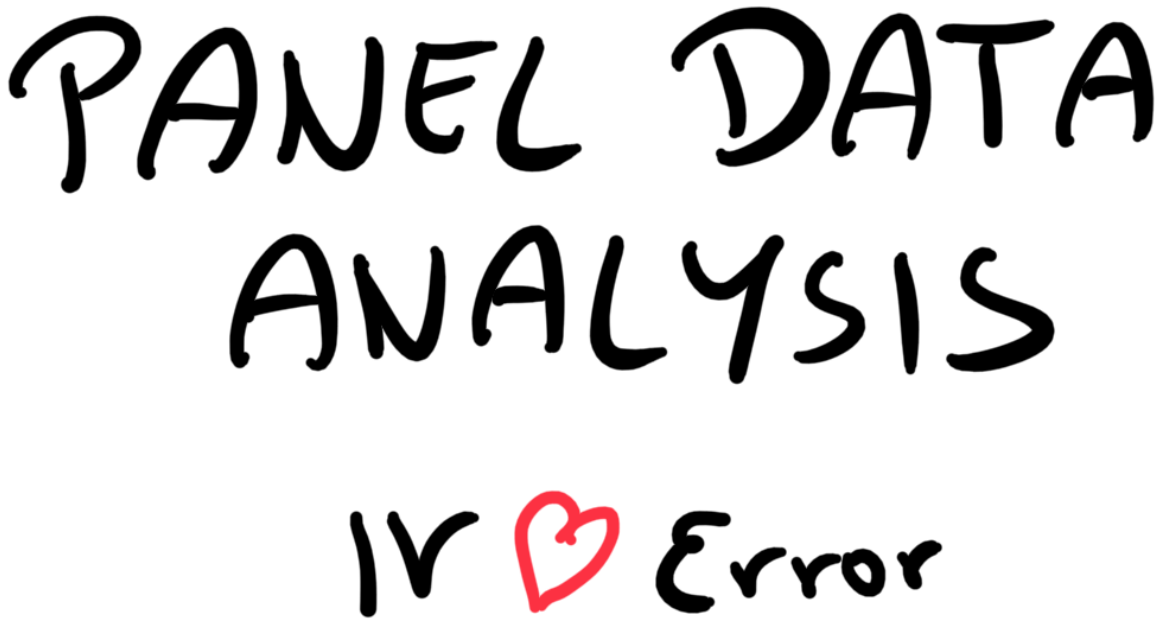


Image by Author

GETTING STARTED

A Guide to Panel Data Regression: Theoretics and Implementation with Python.

Bernhard Brugger Jan 6 · 12 min read

traditional linear regression models. In this article, I want to share the most important theoretics behind this topic and how to build a panel data regression model with Python in a step-by-step manner.

My intention to write this post is twofold: First, in my opinion, it is hard to find an easy and comprehensible explanation of an integrated panel data regression model. Second, performing panel data regression in Python is not as straightforward as in R for example, which doesn´t mean that it is less effective. So, I decided to share my knowledge gained during a recent project in order to make future panel data analysis maybe a bit easier ;-)

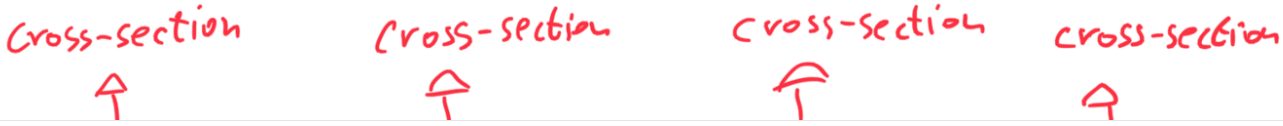
Enough talk! Let´s dive into the topic by describing what panel data is and why it is so powerful!

What is Panel Data?

> *"Panel data is a two-dimensional concept, where the same individuums are observered repeatedly over different periods in time."*

In general, panel data can be seen as a combination of cross-sectional and time-series data. Cross-sectional data is described as one observation of multiple objects and corresponding variables at a specific point in time (i.e. an observation is taken once). Time-series data only observes one object recurrently over time. Panel data comprises characteristics of both into one model by collecting data from multiple, same objects over time.

In a nutshell, we can think of it like a timeline in which we **periodically observe the same individuums**.



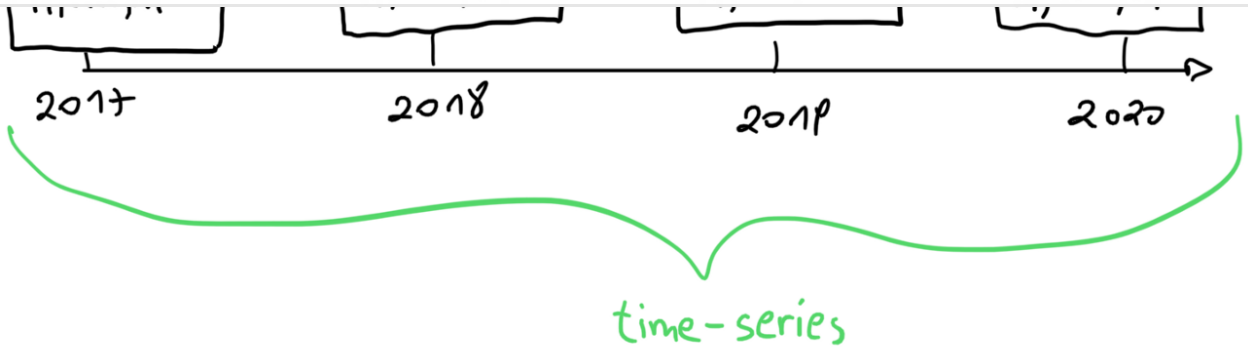


Illustration of the Panel Data Design

Let´s go a step further by breaking down the definition above and explain it stept-by-step on a sample panel dataset:

person	year	x	y
A	2018	3,5	85
A	2019	3,2	83
A	2020	3,8	88
B	2018	1,2	79
B	2019	1,5	83
B	2020	2,3	88
C	2018	5,6	75
C	2019	6	72
C	2020	5,8	78

Sample Panel Dataset

"Panel data is a two-dimensional concept […]": Panel data is commonly stored in a two-dimensional way with rows and columns (we have a dataset with nine rows and four columns). It is important to note that we always need one column to identify the indiviuums under obervation (column *person*) and one column to document the points in time the data was collected (column *year*). Those two columns should be seen as multi-index.

"[…] where the same individuums […]": We have the individuums *person A, person B, and person C,* from which we collect the variables *x* and *y*. The individuums and the observed variables will always stay the same.

sectional data over time, the main difference is that panel data always observes the same individuums, while this cannot be proven in pooled-cross sections.

Example of pooled-cross sections:

year	x	y
2018	3,5	85
2019	3,2	83
2020	3,8	88
2018	1,2	79
2019	1,5	83
2020	2,3	88
2018	5,6	75
2019	6	72
2020	5,8	78

Pooled cross-sectional data

"[…]* are observered repeatedly over different periods in time."*: We collect data from *2018, 2019, and 2020*.

So far so good…now we understand what panel data is. But what is the meaning behind this data concept and why should we use it???

The answer is….**heterogeneity and resulting endogeneity!** Maybe you already heard about this issue in traditional linear regression models, in which heterogeneity often leads to biased results. Panel data is able to deal with that problem.

Since heterogeneity and endogeneity are crucial for understanding why we use panel data models, I will try to explain this problem straightforward in the next section.

The Problem of Endogeneity caused by unobserved Heterogeneity

> *"The unobserved dependency of other independent variable(s) is called* ***unobserved heterogeneity*** *and the correlation between the independent variable(s) and the error term*

Let´s say, we want to analyze the relationship on how coffee consumption affects the level of concentration. A simple linear regression model would look like this:

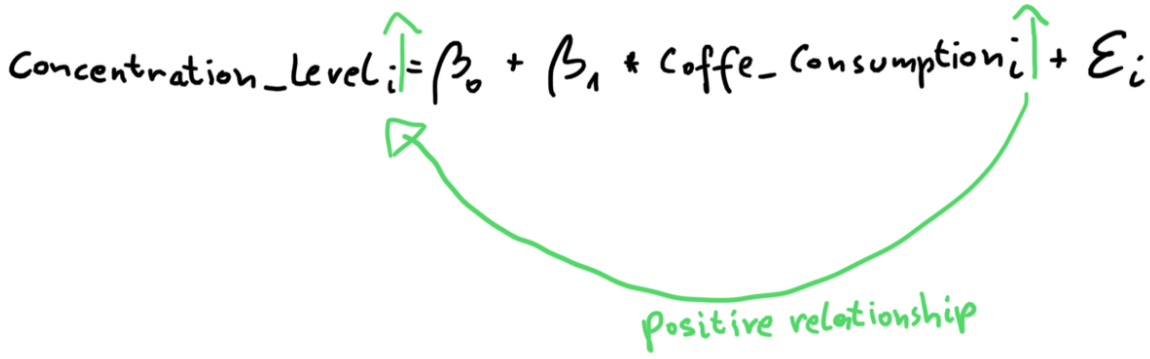
$$Concentration\_Level_i = \beta_0 + \beta_1 * Coffe\_Consumption_i + \varepsilon_i$$

Simple Linear Regression

where:

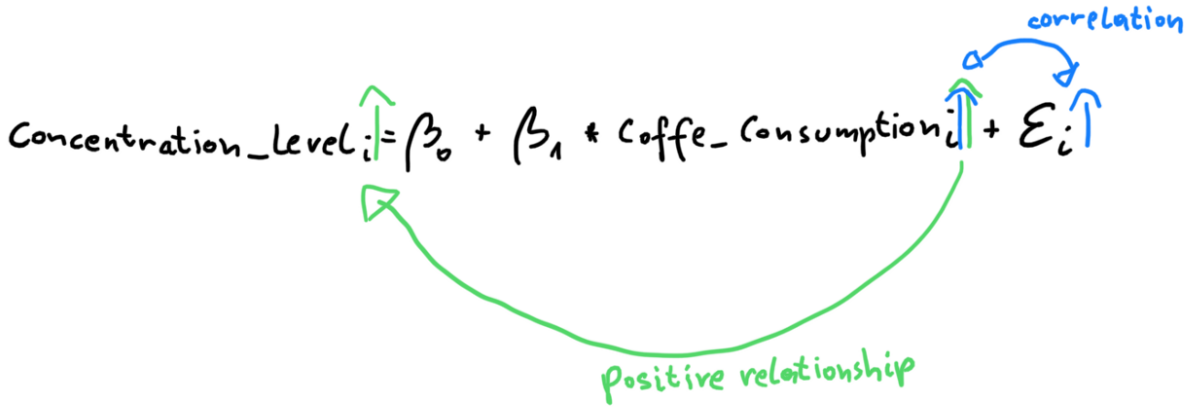
- *Concentration_Level is* the dependent variable (DV)
- *β0* is the intercept
- *β1* is the regression coefficient
- *Coffee_Consumption* is the independent variable (IV)
- ε is the error term

However, the goal of this model is to explore the relationship of *Coffee_Consumption* (IV) on the *Concentration_Level* (DV). Assuming that IV and DV are positively correlated, this would mean that if IV increases, DV would also increase. Let´s add this fact to our formula:



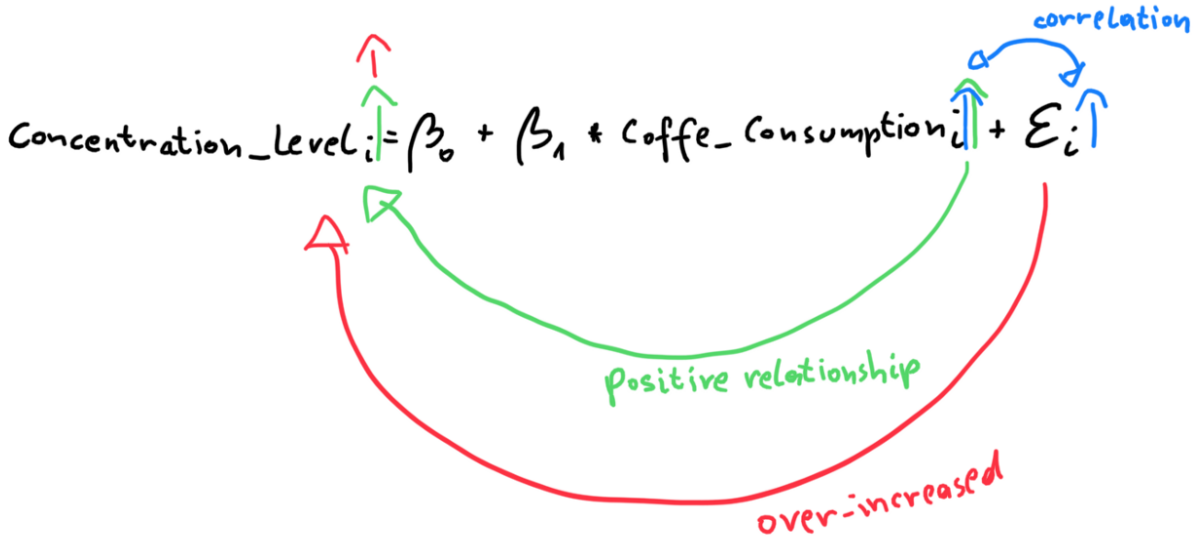
Relationship between IV and DV

you are tired, you will obviously drink coffee ;-)). If you remeber the first sentence of this article, such variables are called unobserved, independent variables. They are "hidden" behind the error term and if, e.g., *Coffee_Consumption* is positively related to such a variable, the error term would increase as *Coffee_Consumption* increases:



Correlation between IV and error term

This, in turn, would lead to an over-increased estimator of the DV *Concentration_Level*. Therefore, the estimated DV is biased and will lead to inaccurate inferences. In our example, the bias would be the red over-increase at *Concentration_Level*.



Luckily, there is a way to deal with this problem…maybe you already guessed it, **panel data regression!** The advantage of panel data is that we can control **heterogeneity** in our regression model by acknowledge heterogeneity as **fix** or **random**. But more on that in the next section!

Types of Panel Data Regression

The following explanations are built on this notation:

$$y_{it} = X_{it}\beta + \alpha_i + u_{it} \quad \text{for } t = 1, \dots, T \text{ and } i = 1, \dots, N$$

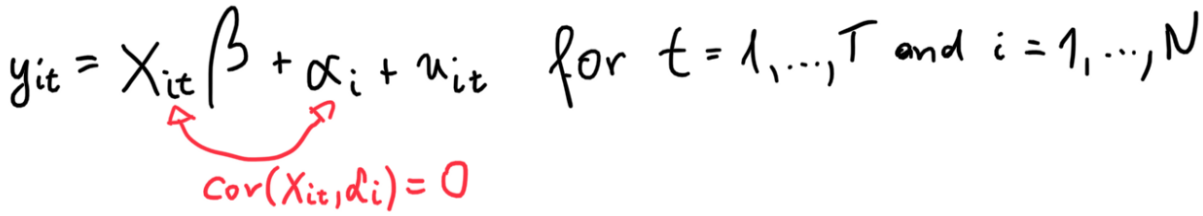
Notation

where:

- y = DV
- X = IV(s)
- β = Coefficients
- α = Individual Effects
- μ = Idiosyncratic Error

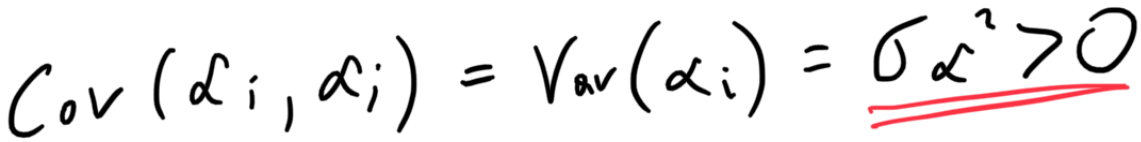
Basically, there are three types of regression for panel data:

1) *PooledOLS:* *PooledOLS* can be described as simple OLS (Ordinary Least Squared) model that is performed on panel data. It ignores time and individual characteristics and focuses only on dependencies **between** the individuums. However, simple OLS requires



Exogeneity Assumption

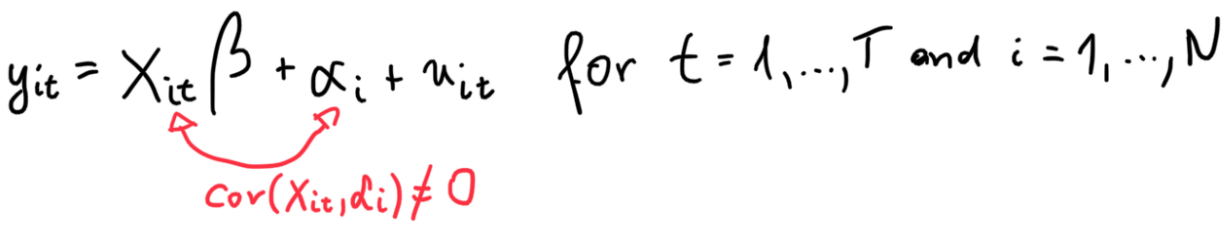
The problem with *PooledOLS* is that even the assumption above holds true, *alpha* might have a serial correlation over time. Consequentely, *PooledOLS* is mostly inappropriate for panel data.



Serial Correlation between alpha

Note: To counter this problem, there is another regression model called *FGLS* (Feasible Generalized Least Squares), which is also used in random effects models described below.

2) *Fixed-Effects (FE) Model:* The FE-model determines individual effects of unobserved, independent variables as constant ("fix") over time. Within FE-models, the relationship between unobserved, independent variables and the IVs (i.e. endogeneity) can be existent:



The trick in a FE-model is, if we assume *alpha* as constant and subtract the mean values from each equation term, *alpha* (i.e. the unobserved heterogeneity) will get zero and can therefore be neglected:

$$y_{it} - \overline{y_i} = \beta(X_{it} - \overline{X_i}) + \underbrace{(\alpha_i - \overline{\alpha_i})}_{=0} + (u_{it} - \overline{u_i})$$

Get rid of Individual Effects in the FE-Model

Solely, the idiosyncratic error (represented by *my* = unobserved factors that change over time and across units) remains and has to be exogen and non-collinear.

However, because heterogeneity can be controlled, this model allows heterogeneity to be existent within the model. Unfortunately, due to the fact that individual effects are fixed, dependencies can only be observed **within** the individuums.

Note: An alternative to the FE-model is the *LSDV*-model (Least Squares Dummy Variables), in which the (fixed) individual effects are represented by dummy variables. This model will lead to the exact same results, but has a main disadvantage, since it will need a lot more computation power if the regression model is big.

3) *Random-Effects (RE) Model:* RE-models determine individual effects of unobserved, independent variables as random variables over time. They are able to "switch" between OLS and FE and hence, can focus on both, dependencies **between** and **within individuals**. The idea behind RE-models is the following:

Let´s say, we have the same notation as above:

Notation

In order to include between- as well as within-estimators, we first need to define, when to use which estimator. In general, if the covariance between *alpha* and *IV(s)* is zero (or very small), there is no correlation between them and an OLS-model is preferred. If that covariance is not zero, there is a relationship that should be eliminated by using a FE-model:

$$Cov(\alpha_i, X_{it}) \neq 0 \Rightarrow \text{FE-model}$$
$$Cov(\alpha_i, X_{it}) = 0 \Rightarrow \text{OLS}$$

When to use which model?

The problem with using OLS, as stated above, is the serial correlation between *alpha* over time. Hence, RE-models determine which model to take according to the serial correlation of the error terms. To do so, the model uses the term *lambda*. In short, *lambda* calculates how big the variance of *alpha* is. If it is zero, then there will be no variance of *alpha*, which, in turn, means that PooledOLS is the preferred choice. On the other side, if the variance of *alpha* tend to become very big, *lambda* tends to become one and therefore it might make sense to eliminate *alpha* and go with the FE-model.

$$\lambda = 1 - \left(\frac{\sigma_u^2}{\sigma_u^2 + T \cdot \sigma_\alpha^2} \right) \qquad \lambda = 0 \Rightarrow \text{OLS} \qquad \lambda = 1 \Rightarrow \text{FE}$$

Now that we know the common models, how do we decide which model to take? Let´s have a look on that…

How to decide which Model is appropriate?

Choosing between PooledOLS and FE/RE: Basically, there are five assumptions for simple linear regression models that must be fulfilled. Two of them can help us in choosing between PooledOLS and FE/RE.

These assumptions are (1) Linearity, (2) Exogeneity, (3a) Homoskedasticity and (3b) Non-autocorrelation, (4) Independent variables are not Stochastic and (5) No Multicolinearity.

If assumption **(2)** or **(3)** (or both) are violated, then FE or RE might be more suitable.

Choosing between FE and RE: Answering this question depends on your assumption, if the individual, unobserved heterogeneity is a constant or random effect. But this question can also be answered perfoming the Hausman-Test.

Hausman-Test: In simple termns, the Hausman-Test is a test of endogeneity. By running the Hausman-Test, the null hypothesis is that the covariance between IV(s) and *alpha* is zero. If this is the case, then RE is preferred over FE. If the null hypothesis is not true, we must go with the FE-model.

So, we now understand the theoretics behind panel data regression. Let´s go to the fun stuff and build the model in Python step-by-step:

Implementing Panel Data Model in Python

I will use the "Guns.csv" dataset, which is normally provided in R. As stated in the description of this dataset: "Guns is a balanced panel of data on 50 US states, plus the District of Columbia (for a total of 51 states), by year for 1977–1999." (Note: a panel dataset is called "balanced" if there are no missing values within the dataset, otherwise, it would be called "unbalanced").

For terms of simplicity, I will only use the following columns provided by the dataset:

- *State:* This column represents our individuums under observation.
- *Year:* The column Year documents our periodically collected data (between 1977–1999).
- *Income:* Income is our IV and is represented as the per capita personal income.
- *Violent:* Violent is our DV and includes violent crime rates (incidents/ 100,000 inhabitants).

Our "research" question would be: ***How does the income affects crime rate?***

```
# Import and preprocess data
import pandas as pd

dataset = pd.read_csv('Guns.csv', usecols = ['state', 'year',
'income', 'violent'],\
 index_col = ['state', 'year'])

years = dataset.index.get_level_values('year').to_list()
dataset['year'] = pd.Categorical(years)
```

Step 2: Start with PooledOLS and check required assumptions

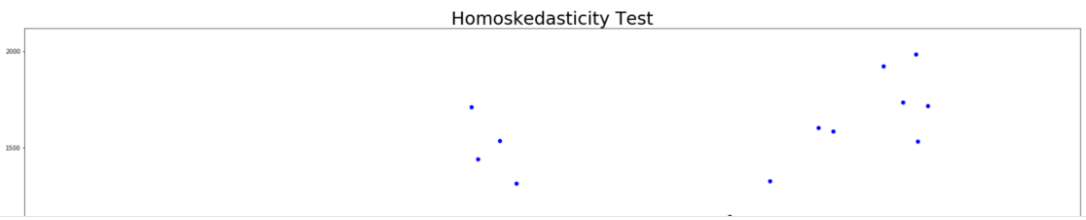
I would recommend to start performing PooledOLS. Since it can be seen as a simple OLS model, it has to fulfill certain assumptions (those in the chapter "How to decide which Model is appropriate?"). As stated above, if condition 2 or 3 (or both) are violated, then FE-/RE-models are likely more suitable. Since condition 2 can only be tested further down with the Hausman-Test, we will stick to proving condition 3 for now.

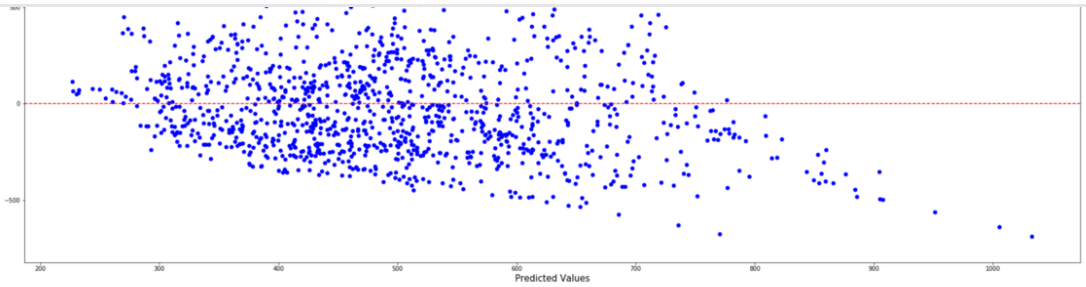
```
# Perform PooledOLS
from linearmodels import PooledOLS
import statsmodels.api as sm

exog = sm.tools.tools.add_constant(dataset['income'])
endog = dataset['violent']
mod = PooledOLS(endog, exog)
pooledOLS_res = mod.fit(cov_type='clustered', cluster_entity=True)

# Store values for checking homoskedasticity graphically
fittedvals_pooled_OLS = pooledOLS_res.predict().fitted_values
residuals_pooled_OLS = pooledOLS_res.resids
```

Check condition 3:

Condition 3 is splitted in 3a (Homoskedasticity) and 3b (Non-Autocorrelation). Those assumptions can be tested with a number of different tests. For condition 3a, I will show you how to identify heteroscedasticity graphically as well as perform the *White-Test* and *Breusch-Pagan-Test* (both are similar). For condition 3b, I will show you the *Durbin-Watson-Test*.

```
# 3A. Homoskedasticity
import matplotlib.pyplot as plt
 # 3A.1 Residuals-Plot for growing Variance Detection
fig, ax = plt.subplots()
ax.scatter(fittedvals_pooled_OLS, residuals_pooled_OLS, color =
‘blue’)
ax.axhline(0, color = 'r', ls = '--')
ax.set_xlabel(‘Predicted Values’, fontsize = 15)
ax.set_ylabel(‘Residuals’, fontsize = 15)
ax.set_title(‘Homoskedasticity Test’, fontsize = 30)
plt.show()
```





Residuals plot for Heteroskedasticity

Basically, a residuals-plot represents predicted values (x-axis) vs. residuals (y-axis). If the plotted data points spread out, this is an indicator for growing variance and thus, for heteroskedasticity. Since this seems to be the case in our example, we might have the first violation. But let´s check this with the White- and the Breusch-Pagan-Test:

```
# 3A.2 White-Test
from statsmodels.stats.diagnostic import het_white, het_breuschpagan

pooled_OLS_dataset = pd.concat([dataset, residuals_pooled_OLS],
axis=1)
pooled_OLS_dataset = pooled_OLS_dataset.drop(['year'], axis =
1).fillna(0)
exog = sm.tools.tools.add_constant(dataset['income']).fillna(0)

white_test_results = het_white(pooled_OLS_dataset['residual'], exog)

labels = ['LM-Stat', 'LM p-val', 'F-Stat', 'F p-val']
print(dict(zip(labels, white_test_results)))

# 3A.3 Breusch-Pagan-Test
breusch_pagan_test_results =
het_breuschpagan(pooled_OLS_dataset['residual'], exog)
labels = ['LM-Stat', 'LM p-val', 'F-Stat', 'F p-val']
print(dict(zip(labels, breusch_pagan_test_results)))
```

In simple terms, if $p < 0.05$, then heteroskedasticity is indicated. Both tests give very small p-values (White-test: **3.442621728589391e-44**, Breusch-Pagan-test: **6.032616972194746e-26**).

```
# 3.B Non-Autocorrelation
# Durbin-Watson-Test
from statsmodels.stats.stattools import durbin_watson

durbin_watson_test_results =
durbin_watson(pooled_OLS_dataset[‘residual’])
print(durbin_watson_test_results)
```

The Durbin-Watson-Test will have one output between 0 – 4. The mean (= 2) would indicate that there is no autocorrelation identified, 0 – 2 means positive autocorrelation (the nearer to zero the higher the correlation), and 2 – 4 means negative autocorrelation (the nearer to four the higher the correlation). In our example, the result is **0.08937264851640213**, which clearly indicates strong positive autocorrelation.

As a consequence, assumption 3b is also violated, so it seems that a FE-/RE-model will be more suitable.

So, let´s build the models!

Step 3: Perform FE- and RE-model

```
# FE und RE model
from linearmodels import PanelOLS
from linearmodels import RandomEffects

exog = sm.tools.tools.add_constant(dataset['income'])
endog = dataset[‘violent’]
# random effects model
model_re = RandomEffects(endog, exog)
re_res = model_re.fit()
# fixed effects model
model_fe = PanelOLS(endog, exog, entity_effects = True)
fe_res = model_fe.fit()
#print results
print(re_res)
print(fe_res)
```

Results FE-model:

```
No. Observations:                1173   R-squared (Within):              0.1127
Date:                 Wed, Nov 11 2020   R-squared (Overall):           0.1140
Time:                         12:29:18   Log-likelihood                -7081.9
Cov. Estimator:            Unadjusted
                                        F-statistic:                    142.39
Entities:                          51   P-value                         0.0000
Avg Obs:                       23.000   Distribution:                F(1,1121)
Min Obs:                       23.000
Max Obs:                       23.000   F-statistic (robust):           142.39
                                        P-value                         0.0000
Time periods:                      23   Distribution:                F(1,1121)
Avg Obs:                       51.000
Min Obs:                       51.000
Max Obs:                       51.000

                             Parameter Estimates
==============================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------
const          181.70     27.101     6.7046     0.0000      128.53      234.88
income         0.0234     0.0020     11.933     0.0000      0.0196      0.0273
==============================================================================

F-test for Poolability: 180.75
P-value: 0.0000
Distribution: F(50,1121)
```

FE-model results

Results RE-model:

```
                        RandomEffects Estimation Summary
================================================================================
Dep. Variable:                  violent   R-squared:                      0.1128
Estimator:                RandomEffects   R-squared (Between):            0.1159
No. Observations:                 1173   R-squared (Within):             0.1127
Date:                 Wed, Nov 11 2020   R-squared (Overall):            0.1156
Time:                         12:29:18   Log-likelihood                -7109.8
Cov. Estimator:            Unadjusted
                                        F-statistic:                    148.90
Entities:                          51   P-value                         0.0000
Avg Obs:                       23.000   Distribution:                F(1,1171)
Min Obs:                       23.000
Max Obs:                       23.000   F-statistic (robust):           148.90
                                        P-value                         0.0000
Time periods:                      23   Distribution:                F(1,1171)
Avg Obs:                       51.000
Min Obs:                       51.000
Max Obs:                       51.000

                             Parameter Estimates
==============================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
```

RE-model results

In this example, both perform similar (although, FE seems to perform slightly better). So, in order to test which model should be preferred, we will finally perfom the Hausman-test.

Step 4: Perform Hausman-Test

Note: Since I had problems with the hausman-function provided in econtools package (covariance was not working), I slightly changed the function. So, you are welcome to use this function, if you are following this guideline.

```
import numpy.linalg as la
from scipy import stats
import numpy as np

def hausman(fe, re):
 b = fe.params
 B = re.params
 v_b = fe.cov
 v_B = re.cov

df = b[np.abs(b) < 1e8].size

chi2 = np.dot((b - B).T, la.inv(v_b - v_B).dot(b - B))

 pval = stats.chi2.sf(chi2, df)

return chi2, df, pval

hausman_results = hausman(fe_res, re_res)
print('chi-Squared: ' + str(hausman_results[0]))
print('degrees of freedom: ' + str(hausman_results[1]))
print('p-Value: ' + str(hausman_results[2]))
```

Since the p-value is very small (**0.008976136961544689**), the null hypothesis can be rejected. Accordingly, the FE-model seems to be the most suitable, because we clearly have endogeneity in our model.

this is stuff for another article ;-)

I really hope you liked this article and it helps you overcoming the common problems with panel data regression. And of course, please don´t be too over-critical since this is my first post on this platform :-)

Panel Data Regression Econometrics Python Getting Started