Methods For Panel Data

Federico D’Amario and Silvio Di Sanzo
aprile 2025

Questa nota, redatta dai colleghi del gruppo di ricerca quantitativa dell'Ufficio Studi Confcommercio, offre un'ampia panoramica dei modelli più utilizzati per l'analisi econometrica dei dati panel (che contengono informazioni su differenti unità statistiche in diversi punti del tempo). Viene presentata una rassegna ragionata sia dei modelli statici sia degli approcci dinamici. Vengono inoltre affrontati in dettaglio i principali problemi econometrici, tra cui l'endogeneità, la correlazione seriale e l'eterogeneità non osservata, tutte questioni rilevanti per l’identificazione di legami causali affidabili e robusti tra le variabili di interesse.
Un'attenzione particolare è, infatti, riservata alle tecniche di inferenza causale, tra cui l'uso delle variabili strumentali e dei metodi di differenza nelle differenze. Infine, i colleghi forniscono i codici Python necessari a replicare autonomamente le tecniche illustrate. Complessivamente, questo lavoro costituisce una lettura preziosa per chiunque voglia capire le relazioni economiche partendo dai dati.

Mariano Bella
Direttore Ufficio Studi Confcommercio

Methods For Panel Data

Federico D'Amario* and Silvio Di Sanzo *

*Ufficio Studi Confcommercio

This note offers a comprehensive exploration of panel data methodologies that serves as an invaluable resource for practitioners, policy makers, and academic researchers. It details both the theoretical foundations and practical applications of key econometric models (e.g, fixed effects, random effects, and advanced estimators such as within, between, and GMM) equipping readers with the necessary tools to handle datasets with both cross-sectional and time-series dimensions. Importantly, each section includes step-by-step practical implementations that serve as a clear guide for readers, enabling them to seamlessly apply the techniques discussed to their own empirical analyses. For practitioners, the detailed discussion and practical implementations provide valuable guidance for applying robust statistical techniques to real-world data. Policy makers can utilize the methodologies presented to inform evidence-based decisions by understanding the dynamic trends and structural nuances embedded in panel data. Meanwhile, academic researchers will appreciate the document's rigorous analytical framework that bridges theoretical advancements with empirical applications, ultimately enhancing the study and practice of modern econometrics. This exploration draws inspiration from and references the work of renowned scholars, including Arellano (2003), Baltagi (2008), Palomba (2008), Angrist and Pischke (2008), Wooldridge (2010), and Hsiao (2014). Their contributions have significantly shaped the methodologies, assumptions, and applications that define this area of study, offering invaluable insights for both theoretical and practical advancements in panel data modelling.

Contents

1 Introduction

Panel data captures information about N individuals at multiple points in time, combining their characteristics at specific moments with repeated observations over T time periods. This dual nature allows panel data to encompass both cross-sectional and time-series elements. In a cross-sectional sense, it provides a snapshot of the attributes of different individuals at a given time. Meanwhile, its time-series dimension enables tracking changes in these attributes across various time intervals.

The data for a variable Y can be organized as a matrix, where each column corresponds to a specific individual, and each row represents observations taken at different points in time. This arrangement results in a total of NT observations for Y, offering a comprehensive dataset that combines individual-specific and temporal variations.

$$Y = \begin{bmatrix} y_{11} & y_{21} & \cdots & y_{i1} & \cdots & y_{N1} \\ y_{12} & y_{22} & \cdots & y_{i2} & \cdots & y_{N2} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ y_{1t} & y_{2t} & \cdots & y_{it} & \cdots & y_{Nt} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ y_{1T} & y_{2T} & \cdots & y_{iT} & \cdots & y_{NT} \end{bmatrix} \tag{1}$$

Analyzing panel data introduces complexities due to the distinct characteristics of cross-sectional and time-series components. These characteristics often result in deviations from the classical assumptions of linear regression. To address this, the relationship between the dependent variable and the regressors is modeled as:

$$Y = X\beta + \varepsilon, \tag{2}$$

In this context, the dependent variable Y is represented as a vector with dimensions $(NT \times 1)$, constructed by stacking the data from the panel matrix into a single column using the `vec` operator. The explanatory variables are organized into the regressor matrix X, which has dimensions $(NT \times k)$. The vector β, of size k, contains the parameters to be estimated, capturing the influence of the regressors on the dependent variable. Finally, ε, a vector of the same size as Y, represents the random disturbances or model errors.

In the context of panel data models, the variance-covariance matrix of the error term, Ω, plays a critical role in understanding the properties of the disturbances. This matrix is symmetric, square, and has dimensions $(NT \times NT)$. It is defined as:

$$\Omega = \text{Var}(\varepsilon) = E(\varepsilon\varepsilon'), \tag{3}$$

The primary advantage of panel data models lies in their ability to increase the efficiency of parameter estimation. The integration of both cross-sectional and time-series dimensions results in a substantially larger number of observations compared to using either dimension in isolation. This increase in sample size directly reduces the variance of the estimator, leading to more precise and reliable inference.

1.1 Practical Implementation

The Python script provided in this practical implementation how to work with panel data using the Grunfeld investment dataset as an example. The Grunfeld dataset is a classic panel dataset frequently used in econometrics to illustrate the analysis of longitudinal data. It comprises observations on ten large U.S. firms over a period of twenty years, making it a balanced panel—a structure where every firm is observed for the same number of time periods. The dataset typically includes key variables such as the firm identi-

fier, year, and measures like annual investment (inv), alongside additional economic indicators like market value (value) and capital stock (capital). Originally compiled for studying corporate investment behavior, it has become a benchmark example in both academic and practical settings to demonstrate techniques such as fixed effects and random effects models. The code begins by importing the required libraries such as pandas, numpy, matplotlib, seaborn, and specialized modules for panel data analysis. The code loads the dataset from an online source, then inspects its initial structure and converts the 'firm' and 'year' columns to string types to facilitate the creation of a multi-index, which is essential for panel data analysis. It proceeds to visualize the evolution of investment over time for each firm by plotting separate time series for each firm. Additionally, the script computes descriptive statistics (mean, standard deviation, minimum, and maximum investments) grouped by firm, and illustrates the transformation of the dataset into a matrix format using a pivot table that reorganizes the investment data by firm and year.

```
# Introduction to Panel Data in Python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from linearmodels import PanelOLS
from linearmodels.panel import RandomEffects

# Load the Grunfeld investment data (a classic panel dataset)
# This dataset contains investment data for 10 large US firms over 20 years
url = "https://raw.githubusercontent.com/vincentarelbundock/Rdatasets/master/
    csv/plm/Grunfeld.csv"
data = pd.read_csv(url)

# Examine the structure of the dataset
print(data.head())

# Convert to panel data format
data['firm'] = data['firm'].astype(str)
data['year'] = data['year'].astype(str)
data = data.set_index(['firm', 'year'])

# Visualize the panel structure - investment over time by firm
plt.figure(figsize=(12, 8))
for firm in data.index.get_level_values(0).unique():
    subset = data.loc[firm]['inv']
    plt.plot(subset.index, subset.values, label=f'Firm {firm}')

plt.title('Investment Over Time by Firm')
plt.xlabel('Year')
plt.ylabel('Investment')
plt.legend(loc='best')
plt.grid(True)
plt.show()
```

```
# Calculate panel data statistics
stats = pd.DataFrame({
    'Mean': data.groupby('firm')['inv'].mean(),
    'Std-Dev': data.groupby('firm')['inv'].std(),
    'Min': data.groupby('firm')['inv'].min(),
    'Max': data.groupby('firm')['inv'].max(),
})

print("Panel-Data-Statistics-by-Firm:")
print(stats)

# Demonstrate the panel data matrix structure mentioned in the paper
# Create a pivot table to represent the matrix in equation (1)
panel_matrix = data['inv'].unstack(level='firm')
print("\nPanel-Data-Matrix-Structure-(Investment):")
print(panel_matrix.head())
```

---

2 Panel Models

Longitudinal data are often organized in a structure similar to the one depicted in matrix (1), where the number of individuals (N) is typically large, while the time dimension (T) remains comparatively small. An important consideration is that, if the assumptions regarding the variance-covariance matrix Ω and the intercept term (if present) align with those of pooled time series models. For the model specified in equation (2), the classical assumptions required for Ordinary Least Squares (OLS) estimation are as follows:

1. $E(\varepsilon|X) = 0$, meaning that the errors have an expected value of zero conditional on the regressors.
2. The matrix X must have full rank, equal to k, ensuring that the model has a unique solution.
3. $E(X'\varepsilon) = 0$, indicating that the regressors and the errors are uncorrelated.
4. The error term's variance-covariance structure is given by $\Omega = \text{Var}(\varepsilon) = E(\varepsilon\varepsilon') = \sigma^2 I_{NT}$. This assumption, which implies homoscedasticity, encompasses several specific conditions:
    - The variance σ^2 is constant for all individuals and time periods ($\forall i$, $\forall t$),
    - $E(\varepsilon_{it}\varepsilon_{is}) = 0$ for $t \neq s$, implying no correlation between observations for the same individual at different time points,
    - $E(\varepsilon_{it}\varepsilon_{jt}) = 0$ for $i \neq j$, indicating no correlation between observations for different individuals at the same time.

When these assumptions hold, the OLS estimator is BLUE (Best Linear Unbiased Estimator).
This section explores the primary techniques used in panel data analysis, focusing on models and estimators that account for individual-specific and temporal variations. Beginning with the Fixed Effects and Random Effects models, the discussion extends to the Within and Between estimators, Generalized Least Squares (GLS) methods, and Feasible GLS (FGLS) approaches. The section concludes with statistical tests, such as the Hausman and Breusch-Pagan tests, to guide model selection and assess assumptions.

2.1 Fixed Effects Model and Within Estimator

Considering the i-th individual, the fixed effects model is specified as follows:

$$y_i = \alpha_i + X_i\beta + \varepsilon_i, \tag{4}$$

where y_i and ε_i are vectors of dimensions $(T \times 1)$, X_i is a matrix of dimension $(T \times k)$, and β is the parameter vector of length k to be estimated. A key feature of equation (4) is the constant term α_i, which is configured as a vector of T constant elements, each equal to α_i. This implies that for each individual i, only one value of the constant needs to be estimated. If $\alpha_i \neq \alpha_j$ for $i \neq j$, the parameter α_i captures the individual-specific effect, which represents the unique characteristics of each individual that remain unchanged over time.
In practice, the model involves estimating a total of $k + N$ parameters: k elements in the vector β and N individual constants α_i. These individual constants account for the heterogeneity across individuals in the panel, which is a distinctive feature of panel data analysis.
Generalizing equation (4) and rewriting it in matrix form, we obtain:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-1} \\ y_N \end{bmatrix} = \begin{bmatrix} \iota_T & 0 & \cdots & 0 & 0 & X_1 \\ 0 & \iota_T & \cdots & 0 & 0 & X_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \iota_T & 0 & X_{N-1} \\ 0 & 0 & \cdots & 0 & \iota_T & X_N \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \\ \beta \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_{N-1} \\ \varepsilon_N \end{bmatrix}, \tag{5}$$

Let us define ι_T as a vector of ones with T entries. Then, the model can be written concisely as:

$$Y = [(I_N \otimes \iota_T) \quad X] \begin{bmatrix} \alpha \\ \beta \end{bmatrix} + \varepsilon, \tag{6}$$

Alternatively, the model can be represented as:

$$Y = (I_N \otimes \iota_T)\alpha + X\beta + \varepsilon, \tag{7}$$

In this context, Y is a vector of dimension $(NT \times 1)$, $(I_N \otimes \iota_T)$ is a matrix of dimension $(NT \times N)$, X is a matrix of size $(NT \times k)$, and ε is a vector of dimension $(NT \times 1)$.
Since the elements of the vector α are unobservable, they are entirely subsumed into the error term of the model. However, if these unobserved effects are correlated with the explanatory variables X_i, the resulting estimates would be biased.
The model in Equation (7) can still be estimated using OLS because it satisfies all the classical regression assumptions. This approach is commonly referred to as the dummy variable model because it involves the inclusion of N dummy variables (capturing individual effects) in the regressor matrix. The resulting OLS estimator is unbiased, consistent, and efficient (i.e., BLUE). Its explicit analytical form is given by:

$$\begin{bmatrix} \hat{\alpha} \\ \hat{\beta} \end{bmatrix} = \begin{bmatrix} (I_N \otimes \iota_T)'(I_N \otimes \iota_T) & (I_N \otimes \iota_T)'X \\ X'(I_N \otimes \iota_T) & X'X \end{bmatrix}^{-1} \begin{bmatrix} (I_N \otimes \iota_T)'Y \\ X'Y \end{bmatrix}. \tag{8}$$

By leveraging the properties of the Kronecker product, specifically that $(I_N \otimes \iota_T)'(I_N \otimes \iota_T) = I_N \otimes \iota_T'\iota_T = TI_N$, the estimator simplifies to:

$$\begin{bmatrix} \hat{\alpha} \\ \hat{\beta} \end{bmatrix} = \begin{bmatrix} TI_N & (I_N \otimes \iota_T)'X \\ X'(I_N \otimes \iota_T) & X'X \end{bmatrix}^{-1} \begin{bmatrix} (I_N \otimes \iota_T)'Y \\ X'Y \end{bmatrix}. \tag{9}$$

To invert the matrix in the expression above, we use a standard result from partitioned matrices. After performing the necessary algebraic manipulations, the final form of the estimator is obtained as:

$$\begin{bmatrix} \hat{\alpha} \\ \hat{\beta} \end{bmatrix} = \begin{bmatrix} \frac{1}{T}(I_N \otimes \iota_T)'(Y - X\hat{\beta}) \\ (X'MX)^{-1}X'MY \end{bmatrix}, \tag{10}$$

where $M = I_{NT} - P$ is the projection matrix that produces deviations from the temporal arithmetic mean for each individual. This matrix M is square, block-diagonal, symmetric, and idempotent, with dimensions $(NT \times NT)$.

The estimator $\hat{\beta}$, as derived from Equation (10), makes use of the idempotent property of the matrix M[1]. Specifically, it can be expressed as:

$$\hat{\beta} = (X'MX)^{-1}X'MY = (\dot{X}'\dot{X})^{-1}\dot{X}'\dot{Y}, \tag{11}$$

where $\dot{X} = MX$ and $\dot{Y} = MY$ are the transformed variables obtained by removing individual-specific effects. This transformation involves expressing both the dependent variable and the regressors as deviations from their respective individual means over time. The idempotent nature of M ensures that applying this transformation repeatedly does not alter the result.

This estimator is known as the Within Estimator because it relies on within-individual temporal variations to estimate $\hat{\beta}$. By removing individual effects via the matrix M, it effectively uses only the within-group (time-based) variations for estimation. Importantly, the within estimator is numerically identical to the dummy variable estimator, as both account for individual effects, albeit through different approaches.

Once $\hat{\beta}$ is obtained, the excluded individual effects can be retrieved from the residuals. From Equation (7), we have:

$$(I_N \otimes \iota_T)\alpha = Y - X\hat{\beta}. \tag{12}$$

From this, the individual constants $\hat{\alpha}_i$ are derived as follows:

$$\frac{1}{T}(I_N \otimes \iota_T)'(I_N \otimes \iota_T)\alpha = \frac{1}{T}(I_N \otimes \iota_T)'(Y - X\hat{\beta}), \tag{13}$$

$$\hat{\alpha} = \frac{1}{T}(I_N \otimes \iota_T)'(Y - X\hat{\beta}). \tag{14}$$

Equation (14) illustrates that each individual-specific effect, $\hat{\alpha}_i$, can be expressed as:

$$\hat{\alpha}_i = \bar{y}_i - \bar{x}_i\hat{\beta}, \tag{15}$$

where $\bar{y}_i$ and $\bar{x}_i$ represent the individual means of the dependent variable and regressors, respectively. Thus, $\hat{\alpha}_i$ captures effects that vary across individuals but remain constant over time. The within estimator excludes these effects from $\hat{\beta}$ by focusing only on deviations from individual means.

One significant limitation of this method is its inability to estimate coefficients for regressors that are constant within individuals over time. Such regressors are collinear with $(I_N \otimes \iota_T)$ in the original model, as shown in Equation (7). Algebraically, when these variables are transformed using M, the result is a column of zeros in the regressor matrix, causing it to lose rank. Consequently, OLS becomes inapplicable in such cases.

To test for the absence of heterogeneity among individuals, a t-test on the individual constants α_i is inappropriate. Instead, an F-test can be conducted with the null hypothesis $H_0 : \alpha_1 = \alpha_2 = \cdots = \alpha_N$ (imposing $N - 1$ constraints). The test statistic is:

$$\frac{\tilde{\varepsilon}'\tilde{\varepsilon} - \hat{\varepsilon}'\hat{\varepsilon}}{N-1} \cdot \frac{NT - N - k - 1}{\hat{\varepsilon}'\hat{\varepsilon}} \sim F_{N-1,NT-N-k-1}, \tag{16}$$

[1] A matrix M is idempotent if $M^2 = M$. This property is central to projection matrices, like M, which are used to isolate deviations from individual means.

where $\tilde{\varepsilon}$ and $\hat{\varepsilon}$ represent the residuals from the restricted and unrestricted models, respectively. The variance estimator is given by:

$$\hat{\sigma}_{\varepsilon}^2 = \frac{\hat{\varepsilon}'\hat{\varepsilon}}{NT - N - k - 1}. \tag{17}$$

The variance of $\hat{\beta}$ is then:

$$\text{Var}(\hat{\beta}) = \hat{\sigma}_{\varepsilon}^2 (X'MX)^{-1}. \tag{18}$$

Under H_0, the within estimator coincides with the pooled estimator. Moreover, the within estimator is BLUE, consistent as $NT \to \infty$ and asymptotically normal:

$$\sqrt{NT}(\hat{\beta} - \beta) \xrightarrow{d} N\left(0, \sigma_{\varepsilon}^2 Q^{-1}\right), \tag{19}$$

where $Q = \lim_{NT\to\infty} \frac{1}{NT} X'MX$.

2.1.1 Practical Implementation

In this practical example we show hot to implement a fixed effects panel data model using the Grunfeld investment dataset. The code begins by importing necessary libraries and loading the dataset, then structures it for panel analysis by setting firm and year as indices. The core analysis estimates a fixed effects model where investment is a function of firm value and capital stock, capturing unobserved firm-specific heterogeneity through entity effects. After running the PanelOLS regression, the code extracts the estimated fixed effects (individual firm intercepts) and demonstrates how these can be manually calculated using group means and estimated coefficients—essentially showing the mathematical relationship behind the Within estimator. It then visualizes these fixed effects through a bar chart and conducts an F-test to determine whether the heterogeneity across firms is statistically significant, effectively testing whether pooled OLS would be appropriate or if the fixed effects specification is necessary.

```
# Fixed Effects Model and Within Estimator
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from linearmodels import PanelOLS

# Load the Grunfeld dataset
url = "https://raw.githubusercontent.com/vincentarelbundock/Rdatasets/master/
    csv/plm/Grunfeld.csv"
data = pd.read_csv(url)

# Prepare data for panel analysis
data = data.set_index(['firm', 'year'])

# Specify the model: investment as a function of value and capital
# This corresponds to equation (4)
formula = 'inv ~ value + capital + EntityEffects'
```

```
# Estimate the fixed effects model (Within estimator)
mod = PanelOLS.from_formula(formula, data)
fe_results = mod.fit()

print("Fixed Effects (Within) Estimator Results:")
print(fe_results.summary)

# Calculate and examine the fixed effects (individual constants  _i )
# This corresponds to equation (15)
fixed_effects = fe_results.estimated_effects
print("\nEstimated Fixed Effects (  _i  ) for each firm:")
print(fixed_effects)

# Compare with manual calculation of fixed effects
# Get the coefficients
beta_value = fe_results.params['value']
beta_capital = fe_results.params['capital']

# Calculate individual means
means = data.groupby(level='firm').mean()

# Manually calculate fixed effects using equation (25)
manual_fixed_effects = means['inv'] - (beta_value * means['value'] +
    beta_capital * means['capital'])
print("\nManually Calculated Fixed Effects:")
print(manual_fixed_effects)

# Check the structure of fixed_effects
print("Type:", type(fixed_effects))
print("Shape:", fixed_effects.shape)
print("First few values:")
print(fixed_effects.head())

# Alternative visualization
plt.figure(figsize=(10, 6))
plt.bar(range(len(fixed_effects)), fixed_effects.values.flatten())
plt.xticks(range(len(fixed_effects)), fixed_effects.index.get_level_values(0))
plt.title('Fixed Effects by Firm')
plt.xlabel('Firm')
plt.ylabel('Fixed Effect (  _i  )')
plt.grid(axis='y')
plt.show()

# Test for the absence of heterogeneity (F-test)
# Compare pooled OLS with fixed effects model
pooled_formula = 'inv ~ value + capital'
pooled_mod = PanelOLS.from_formula(pooled_formula, data)
```

```
pooled_results = pooled_mod.fit()


# F-test for fixed effects (corresponds to equation 26)
f_stat = fe_results.f_statistic_robust.stat
f_pval = fe_results.f_statistic_robust.pval
print(f"F-test for fixed effects: F-statistic = {f_stat:.4f}, p-value = {
    f_pval:.4f}")
if f_pval < 0.05:
    print("Conclusion: Reject the null hypothesis. Individual fixed effects
        are significant.")
else:
    print("Conclusion: Fail to reject the null hypothesis. No evidence of
        individual heterogeneity.")
```

2.2 Random Effects Model and Betweeen Estimator

The random effects model assumes that individual-specific effects are stochastic components of the error term, uncorrelated with the regressors. Unlike the fixed effects model, this approach allows the inclusion of variables in X that vary across individuals but remain constant within the T observations for each individual. For the i-th individual, the random effects model is expressed as:

$$y_i = \alpha_i + X_i\beta + \varepsilon_i = \alpha + X_i\beta + \mu_i + \varepsilon_i, \tag{20}$$

where $\alpha_i = \alpha + \mu_i$ consists of a fixed component, α, common to all individuals, and a random component, μ_i, which varies across individuals. For a given i, α_i is a vector of constants of dimension $(T \times 1)$.
A key condition for consistent estimation in this model is the absence of correlation between α_i and the regressor matrix X_i for all individuals.
In the random effects model, the error term ε_i shares the same properties as in the fixed effects model, but additional assumptions are imposed on the individual-specific component μ_i:

1. $E(\mu_i) = 0$,
2. $\text{Var}(\mu_i) = \sigma_\mu^2$ for all $i = 1, 2, \ldots, N$,
3. $E(\mu_i, \mu_j) = 0$ for all $i \neq j$ (no correlation between individual effects),
4. $E(\mu_i, \varepsilon_{j,t}) = 0$ for all i, j, t (no correlation between individual effects and disturbances).

The model can be expressed compactly as:

$$Y = \alpha + X\beta + (\mu \otimes \iota_T) + \varepsilon, \tag{21}$$

where μ is an N-dimensional vector of individual effects. Defining $U = (\mu \otimes \iota_T) + \varepsilon$, the total error in the random effects model consists of a component that varies across individuals but is constant over time and another that varies randomly across individuals and over time.
Under the assumptions outlined earlier, the variance-covariance matrix of U is central to the random effects model and is defined as:

$$\Omega = E(UU') = E\{[(\mu \otimes \iota_T) + \varepsilon][(\mu \otimes \iota_T) + \varepsilon]'\} = E[(\mu \otimes \iota_T)(\mu \otimes \iota_T)' + \varepsilon\varepsilon'] = E(\mu\mu' \otimes \iota_T\iota_T' + \varepsilon\varepsilon'). \quad (22)$$

Given that $E(\mu\mu') = \sigma_\mu^2 I_N$, the term $E(\mu\mu' \otimes \iota_T\iota_T')$ takes a block diagonal form. Additionally, since $E(\varepsilon\varepsilon') = \sigma_\varepsilon^2 I_{NT}$, we can write:

$$\Omega = \sigma_\mu^2(I_N \otimes \iota_T\iota_T') + \sigma_\varepsilon^2 I_{NT} = I_N \otimes (\sigma_\mu^2 \iota_T\iota_T' + \sigma_\varepsilon^2 I_T). \quad (23)$$

The matrix Ω is block diagonal, with each block given by:

$$\Omega_i = \begin{bmatrix} \sigma_\mu^2 + \sigma_\varepsilon^2 & \sigma_\mu^2 & \cdots & \sigma_\mu^2 \\ \sigma_\mu^2 & \sigma_\mu^2 + \sigma_\varepsilon^2 & \cdots & \sigma_\mu^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_\mu^2 & \sigma_\mu^2 & \cdots & \sigma_\mu^2 + \sigma_\varepsilon^2 \end{bmatrix}_{(T\times T)} . \quad (24)$$

The composite error U exhibits constant autocorrelation over time, with a uniform variance-covariance structure across all individuals in the panel.
Since this variance-covariance matrix is block diagonal, the random effects model must be estimated using the GLS method, thus

$$\hat{b} = (X'\Omega^{-1}X)^{-1}X'\Omega^{-1}Y, \quad (25)$$

where $\hat{b} = [\hat{\alpha}, \hat{\beta}]'$ has dimension $(k+1)$. The inverse matrix Ω^{-1} is given by

$$\Omega^{-1} = (I_N \otimes \Omega_i)^{-1} = I_N \otimes \Omega_i^{-1} = I_N \otimes (\sigma_\mu^2 \iota_T\iota_T' + \sigma_\varepsilon^2 I_T)^{-1}. \quad (26)$$

Let's introduce now the matrices P and M. The matrix P (Projection Matrix) is defined as $P = (I_N \otimes P_\iota)$ where $P_\iota = \iota_T(\iota_T'\iota_T)^{-1}\iota_T'$, and is a square, block diagonal, symmetric and idempotent matrix of dimension $(NT \times NT)$ that, when multiplied with panel data matrix X, produces individual means ($PX = \bar{X}$) and geometrically represents orthogonal projections onto space generated by ι_T. The matrix M (Deviation Matrix) is defined as $M = (I_N \otimes M_\iota)$ where $M_\iota = I_T - P_\iota = I_T - \iota_T(\iota_T'\iota_T)^{-1}\iota_T'$, sharing similar properties of being square, block diagonal, symmetric and idempotent, but producing deviations from individual means ($MX = X - \bar{X}$) and representing distances between vectors and their orthogonal projections. These matrices have key relationships: $P + M = I_{NT}$, $PM = 0$, $\iota_T'M_\iota = M_\iota\iota_T = 0$, $\iota_T'P_\iota = P_\iota\iota_T = \iota_T$, and for scalars c_1 and c_2 (which are arbitrary real number constants used in linear combinations): $(c_1P + c_2M)^s = c_1^sP + c_2^sM$, $(c_1P + c_2M)^{-1} = \frac{1}{c_1}P + \frac{1}{c_2}M$, and the quadratic form $(c_1P + c_2M)'(c_1P + c_2M) = c_1^2P + c_2^2M$. Adding and subtracting $P_\iota\sigma_\varepsilon^2$, we get

$$\Omega^{-1} = I_N \otimes \left[(T\sigma_\mu^2 + \sigma_\varepsilon^2)P_\iota + \sigma_\varepsilon^2(I_T - P_\iota)\right]^{-1} = I_N \otimes \left[(T\sigma_\mu^2 + \sigma_\varepsilon^2)P + \sigma_\varepsilon^2 M\right]^{-1} = \left[(T\sigma_\mu^2 + \sigma_\varepsilon^2)P + \sigma_\varepsilon^2 M\right]^{-1}$$

Letting $\sigma^2 = (T\sigma_\mu^2 + \sigma_\varepsilon^2)$, by the properties of the matrices P and M, we have

$$\Omega^{-1} = \frac{1}{\sigma^2}P + \frac{1}{\sigma_\varepsilon^2}M, \quad (27)$$

and thus

$$\Omega^{-1/2} = \frac{1}{\sigma}P + \frac{1}{\sigma_\varepsilon}M. \quad (28)$$

From this definition, it follows that the GLS estimator for the random effects model coincides with the OLS estimator of the regression of $\dot{Y} = \Omega^{-1/2}Y$ on $\dot{X} = \Omega^{-1/2}X$. The properties of this estimator are:

1. If σ_ε^2 and σ_μ^2 are known, the GLS estimator is consistent for $N \to \infty$ and $T \to \infty$,
2. For a given T, the GLS estimator is more efficient than the within estimator; for $N \to \infty$, this efficiency tends to disappear,
3. If $\Omega^{-1} \equiv M$, the GLS estimator coincides with the within estimator, hence the random effects model coincides with the fixed effects model: this can occur if the only source of variability comes from individual effects μ_i. Analytically, it must result that
   - $\sigma_\varepsilon^2 = 0$ (vector ε is constant for every i and t),
   - $T \to \infty$ (by definition $\hat{\sigma}_\varepsilon^2 = 0$): in this case, individual effects become observable.
4. If $\Omega^{-1} \equiv I_{NT}$, the random effects model becomes a standard OLS model and coincides with a pooled time series model; in this case, naturally $\sigma_\mu^2 = 0$, so there are no individual effects, and all variability depends on the disturbance term ε.

The random effects model in equation (21) can be rewritten using the Between transformation, which reformulates the variables as their individual time averages. This transformation focuses on capturing variability between individuals by expressing the variables in terms of their averages over time for each individual. By doing so, the Between estimator isolates cross-sectional variation and provides insights into differences across individuals while disregarding within-individual changes over time. Algebraically, the transformation is implemented by premultiplying the entire equation by the matrix P:

$$PY = P\alpha + PX\beta + P[(\mu \otimes \iota_T) + \varepsilon] = PX\beta + Pu. \tag{29}$$

The estimator used is a GLS, which is implemented as an OLS regression of $\dot{Y} = PY$ on $\dot{X} = PX$. Specifically, we have:

$$\hat{b} = (X'P^{-1}X)^{-1}X'P^{-1}Y = (\dot{X}'\dot{X})^{-1}\dot{X}'\dot{Y}, \tag{30}$$

where $\hat{b} = [\hat{\alpha}, \hat{\beta}]'$ is a vector of dimension $(k+1)$. The estimator defined in equation (30) is unbiased and remains consistent as $N \to \infty$.

The between estimator, like the within estimator, involves a trade-off in terms of information efficiency. By focusing on time-averaged values for each individual, the transformation inherently discards within-individual variation, leading to a loss of efficiency.

While the within estimator leverages the variation within each individual over time (centered around individual means or "within-group" differences), the between estimator captures differences across individuals ("between-group" variation). Essentially, it regresses the cross-sectional means of the dependent variable on the corresponding means of the regressors, reducing the data to a single observation per individual.

For example, in a study examining the relationship between income and education, the within estimator would analyze how changes in education levels affect income for the same individual over time. In contrast, the between estimator would assess whether individuals with higher average education levels tend to have higher average incomes across the population.

2.2.1 Practical Implementation

In this implementation we show how different panel data estimators capture investment behaviour using the Grunfeld dataset, implementing three key approaches side by side. We first estimate a Random Effects

model that balances both within-firm and between-firm variation, treating firm-specific effects as random draws from a distribution. We then implement the Between Estimator, which focuses exclusively on cross-sectional differences by analyzing firm averages over time. For comparison, we include the Fixed Effects (Within) Estimator from our previous analysis, which isolates within-firm temporal variation. Going beyond simple estimation, we decompose error variance into within-group (σ_ε^2) and between-group (σ_μ^2) components, calculating the critical θ parameter that determines how our Random Effects estimator balances between approaches. Through visualizations and coefficient comparisons, we demonstrate how these methods yield different results, and verify our variance component calculations by confirming their relationship with the model's θ parameter. This comprehensive analysis shows whether firm-specific characteristics or temporal changes drive investment patterns, providing practical guidance for econometric model selection.

```
    # Random Effects Model and Between Estimator
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from linearmodels import RandomEffects, BetweenOLS, PanelOLS

# Load the Grunfeld dataset
url = "https://raw.githubusercontent.com/vincentarelbundock/Rdatasets/master/
    csv/plm/Grunfeld.csv"
data = pd.read_csv(url)
data = data.set_index(['firm', 'year'])

# 1. Random Effects Model (Equation 20 in section 2.2)
# y_i = alpha_i + X_i*beta + epsilon_i = alpha + X_i*beta + mu_i + epsilon_i
re_formula = 'inv ~ value + capital'
re_mod = RandomEffects.from_formula(re_formula, data)
re_results = re_mod.fit()

print("Random Effects Model Results:")
print(re_results.summary)

# 2. Between Estimator (Equations 30-31 in section 2.2)
# The between estimator is implemented by regressing individual time averages
between_mod = BetweenOLS.from_formula(re_formula, data)
between_results = between_mod.fit()

print("\nBetween Estimator Results:")
print(between_results.summary)

# 3. Compare Fixed Effects (Within), Random Effects, and Between Estimators
fe_mod = PanelOLS.from_formula('inv ~ value + capital + EntityEffects', data)
fe_results = fe_mod.fit()

# Create a comparison table of coefficients
coef_comparison = pd.DataFrame({
    'Fixed Effects (Within)': fe_results.params,
```

```
    'Random Effects': re_results.params,
    'Between': between_results.params
})

print("\nCoefficient Comparison:")
print(coef_comparison)

# 4. Visualize the comparison
plt.figure(figsize=(10, 6))
coef_comparison.loc[['value', 'capital']].plot(kind='bar')
plt.title('Coefficient Comparison: Fixed Effects vs Random Effects vs Between'
    )
plt.ylabel('Coefficient Value')
plt.grid(axis='y')
plt.show()

# 5. Calculate variance components following section 2.3 (FGLS Estimator)
# Get residuals and group means for variance calculations
T = len(data.index.get_level_values(1).unique())  # Number of time periods
N = len(data.index.get_level_values(0).unique())  # Number of entities

# Get theta from random effects model - extract scalar value from DataFrame
# Print the type and shape to debug
print("Theta type:", type(re_results.theta))
if hasattr(re_results.theta, 'shape'):
    print("Theta shape:", re_results.theta.shape)

# Extract scalar value using appropriate method based on the type
if isinstance(re_results.theta, pd.DataFrame):
    theta_value = re_results.theta.iloc[0, 0]  # Extract first element if
        DataFrame
elif isinstance(re_results.theta, pd.Series):
    theta_value = re_results.theta.iloc[0]  # Extract first element if Series
else:
    theta_value = re_results.theta  # Use as is if already scalar

# Calculate fixed effects residuals for sigma_e^2 (Equation 32 in section 2.3)
# sigma_e^2 represents the within-group variance component
fe_residuals = fe_results.resids
sigma_e_squared = fe_residuals.pow(2).sum() / (N * T - N - 2)

# Get between model residuals for sigma_mu^2 calculation
between_residuals = between_results.resids
sigma_r_squared = between_residuals.pow(2).sum() / (N - 2 - 1)

# Calculate sigma_mu^2 using equation (35) in section 2.3
# sigma_mu^2 = sigma_R^2 - (sigma_e^2 / T)
```

```
if isinstance(sigma_r_squared, pd.Series):
    sigma_r_squared = sigma_r_squared.iloc[0]
if isinstance(sigma_e_squared, pd.Series):
    sigma_e_squared = sigma_e_squared.iloc[0]

sigma_mu_squared = max(0, sigma_r_squared - (sigma_e_squared / T))

# Calculate standard deviations
sigma_mu = np.sqrt(sigma_mu_squared)
sigma_e = np.sqrt(sigma_e_squared)

print(f"\nVariance Components (following equations in section 2.3):")
print(f"Between-group variance component (sigma_mu^2): {sigma_mu_squared:.4f}"
    )
print(f"Within-group variance component (sigma_e^2): {sigma_e_squared:.4f}")
print(f"Between-group std dev (sigma_mu): {sigma_mu:.4f}")
print(f"Within-group std dev (sigma_e): {sigma_e:.4f}")
print(f"Theta (weight parameter): {theta_value:.4f}")

# Calculate the ratio of variances to determine which component dominates
variance_ratio = sigma_mu_squared / sigma_e_squared
print(f"Ratio of between to within variance (sigma_mu^2/sigma_e^2): {
    variance_ratio:.4f}")

print("\nInterpretation based on variance components:")
if variance_ratio > 1:
    print("Between-group variation dominates (sigma_mu^2 > sigma_e^2),
        suggesting the Between estimator might be more efficient.")
else:
    print("Within-group variation dominates (sigma_e^2 > sigma_mu^2),
        suggesting the Within estimator might be more efficient.")

# Verify relations with theta parameter based on equations (27-29) in section
    2.2
print("\nVerification with theta parameter:")
calculated_theta = 1 - (sigma_e / np.sqrt(T * sigma_mu_squared +
    sigma_e_squared))
print(f"Calculated GLS weight: {calculated_theta:.4f}")
print(f"Model's theta parameter: {theta_value:.4f}")
print("The closeness of these values verifies our variance component
    calculations align with the model.")
```

2.3 Feasible GLS Estimator (FGLS)

When the variances σ_ε^2 (idiosyncratic error variance) and σ_μ^2 (individual effect variance) are known, the GLS estimator can be directly applied to efficiently estimate the model parameters. However, in practice, these

variances are typically unobservable, necessitating an alternative approach: the Feasible Generalized Least Squares (FGLS) estimator.

1. Estimating σ_ε^2
   The process begins by estimating σ_ε^2 using the residuals from the within estimator, denoted as $\hat{\varepsilon}_{wit}$. These residuals capture the deviations from the within-group means. The variance is then computed as:

$$\hat{\sigma}_\varepsilon^2 = \frac{\hat{\varepsilon}'_{wit} M \hat{\varepsilon}_{wit}}{NT - N - k}, \tag{31}$$

   where the denominator adjusts for the degrees of freedom lost in estimating $N + k$ parameters.

2. Variance of the Random Effects
   In the random effects model, the variability in the dependent variable is decomposed into two components: the variance of the individual effect (σ_μ^2) and the variance of the error term averaged over time. For an individual i, the model based on time averages is expressed as:

$$y_i - \alpha - \beta x_i = \mu_i + \varepsilon_i,$$

   with the composite term $u_i = \mu_i + \varepsilon_i$. The variance of u_i is given by:

$$\text{Var}(u_i) = \sigma_R^2 = \sigma_\mu^2 + \frac{\sigma_\varepsilon^2}{T}, \tag{32}$$

   where: - σ_R^2 is the total variance of the composite error term u_i, - $\frac{\sigma_\varepsilon^2}{T}$ represents the average contribution of the idiosyncratic error to the overall variance.

3. Estimating σ_R^2 and σ_μ^2
   The residuals $\hat{u}_i$ from the random effects model are used to estimate σ_R^2 as:

$$\hat{\sigma}_R^2 = \frac{\hat{u}'_i \hat{u}_i}{N - k}, \tag{33}$$

   where k accounts for the number of regressors. Using the relationship in equation (32), the individual effect variance σ_μ^2 is then computed as:

$$\hat{\sigma}_\mu^2 = \hat{\sigma}_R^2 - \frac{\hat{\sigma}_\varepsilon^2}{T}. \tag{34}$$

4. Applying the Feasible GLS
   With the estimated variances $\hat{\sigma}_\varepsilon^2$ and $\hat{\sigma}_\mu^2$, the FGLS method can be applied to efficiently estimate the model. This approach adjusts for the heteroskedasticity and correlation induced by the panel structure.

In finite samples, equation (34) might yield a negative value for $\hat{\sigma}_\mu^2$. This typically occurs due to sampling variability or when the variance of the individual effects is small relative to the error variance. In such cases, modifications (e.g., truncating $\hat{\sigma}_\mu^2$ to zero) may be necessary, though they can affect efficiency. The FGLS estimator combines information from both within-individual and between-individual variation, making it suitable for cases where individual effects are assumed to be random and uncorrelated with the regressors. To provide an example imagine analyzing the effect of education on income using panel data. The within estimator looks at how changes in education within an individual (e.g., obtaining a degree) impact their income over time. The between estimator compares individuals with higher average education levels to

those with lower ones. FGLS combines these perspectives while adjusting for the fact that individual-specific effects (e.g., innate ability) and measurement errors differ in their impact across and within individuals. By estimating the variances of these effects, FGLS optimally balances these two sources of variation, improving efficiency compared to simpler methods.

2.3.1 Practical Implementation

In the code below we show the step-by-step implementation of the Feasible Generalized Least Squares estimator for panel data, using the Grunfeld investment dataset as our working example. We begin by estimating a fixed effects model to extract residuals, which we use to calculate σ_ε^2 (the variance of the idiosyncratic error component). Then, we employ the Between estimator to calculate σ_R^2 (the total between-group variance), allowing us to derive σ_μ^2 (the variance of the individual effect) using the relation $\sigma_\mu^2 = \sigma_R^2 - \sigma_\varepsilon^2/T$. With these variance components, we compute the crucial transformation parameter θ, which determines the weight given to between-group variation in our FGLS estimator. We then demonstrate two implementation approaches: a manual transformation where we explicitly demean the data by θ times the group means before running OLS, and a comparison with the built-in RandomEffects estimator from the linearmodels package. This parallel implementation illuminates the mathematical foundations of FGLS for panel data, showing how theoretical variance component calculations translate into practical econometric estimation.

```
  # Feasible GLS Estimator for Panel Data
import pandas as pd
import numpy as np
import statsmodels.api as sm
from linearmodels import PanelOLS, RandomEffects, BetweenOLS
from statsmodels.regression.linear_model import GLS

# Load the Grunfeld dataset
url = "https://raw.githubusercontent.com/vincentarelbundock/Rdatasets/master/
    csv/plm/Grunfeld.csv"
data = pd.read_csv(url)
data = data.set_index(['firm', 'year'])

# Step 1: Estimate the model using the within estimator (fixed effects)
fe_mod = PanelOLS.from_formula('inv ~ value + capital + EntityEffects', data)
fe_results = fe_mod.fit()

# Step 2: Get the residuals from the within estimator
fe_residuals = fe_results.resids

# Step 3: Estimate sigma_e  (variance of idiosyncratic error)
# Using equation (32) in section 2.3 of the document
N = len(data.index.get_level_values(0).unique())  # Number of entities
T = len(data.index.get_level_values(1).unique())  # Number of time periods
k = 2  # Number of regressors (value and capital)

# Calculate variance of idiosyncratic error
sigma_e_squared = fe_residuals.pow(2).sum() / (N * T - N - k)
```

```
print(f"Estimated sigma_e  (idiosyncratic error variance): {sigma_e_squared
    :.4f}")

# Step 4: Estimate Between model to get residuals for sigma_
# Use BetweenOLS from linearmodels instead of manual calculation
between_mod = BetweenOLS.from_formula('inv ~ value + capital', data)
between_results = between_mod.fit()
between_residuals = between_results.resids

# Calculate variance of the random effects model residuals
# Following equation (34) in section 2.3 of the document
sigma_r_squared = between_residuals.pow(2).sum() / (N - k - 1)
print(f"Estimated sigma_R  (total between variance): {sigma_r_squared:.4f}")

# Step 5: Calculate sigma_    (variance of individual effect)
# Using equation (35) in section 2.3 of the document
sigma_mu_squared = sigma_r_squared - (sigma_e_squared / T)
if sigma_mu_squared < 0:
    print("Warning: Calculated sigma_    is negative. Setting to zero.")
    sigma_mu_squared = 0
print(f"Estimated sigma_    (individual effect variance): {sigma_mu_squared
    :.4f}")

# Step 6: Calculate the weight for FGLS transformation
# Based on equations (27-29) in section 2.2 of the document
theta = 1 - np.sqrt(sigma_e_squared / (T * sigma_mu_squared + sigma_e_squared)
    )
print(f"Estimated    (transformation weight): {theta:.4f}")

# Step 7: Compare with RandomEffects implementation
re_mod = RandomEffects.from_formula('inv ~ value + capital', data)
re_results = re_mod.fit()
print(f"\nRandom Effects model    : {re_results.theta}")

# Step 8: Manual FGLS implementation
# Transform the data using the estimated
data_reset = data.reset_index()

# Step 2: Calculate means per firm
firm_means = data_reset.groupby('firm')[['inv', 'value', 'capital']].mean()

# Step 3: Apply transformation
for firm in data_reset['firm'].unique():
    firm_mask = data_reset['firm'] == firm
    for var in ['inv', 'value', 'capital']:
        data_reset.loc[firm_mask, f'{var}_trans'] = (
            data_reset.loc[firm_mask, var] - theta * firm_means.loc[firm, var]
```

```
        )

# Step 4: Use transformed variables for regression
fgls_mod = sm.OLS(data_reset['inv_trans'],
                  sm.add_constant(data_reset[['value_trans', 'capital_trans'
                      ]]))
fgls_results = fgls_mod.fit()

print("\nFeasible GLS Results (Manual Implementation):")
print(fgls_results.summary().tables[1])

print("\nRandom Effects Results (linearmodels implementation):")
print(re_results.summary.tables[1])
```

2.4 Statistical Tests

To decide whether it is preferable to estimate a fixed effects model or a random effects model, some test procedures can be used. The most famous are the Breusch and Pagan (1980) test and the Hausman (1978) test.

2.4.1 Breusch and Pagan Test

The Breusch and Pagan (BP) test is a widely used diagnostic tool for detecting heteroscedasticity in linear regression models. Heteroscedasticity arises when the variance of the error term is not constant across observations, potentially leading to inefficient estimators and incorrect inference. The BP test is applied to models of the form $Y = X\beta + \varepsilon$, where $\varepsilon \sim N(0, \sigma^2\Omega)$. The null hypothesis of the BP test assumes homoscedasticity (constant error variance), while the alternative hypothesis suggests heteroscedasticity.
The variance of the error term can be modeled as:

$$\text{Var}(\varepsilon) = \sigma^2 f(Z\gamma) = \sigma^2 f(\gamma_0 + \gamma_1 Z_1 + \cdots + \gamma_q Z_q), \tag{35}$$

where Z is a matrix containing variables hypothesized to explain the heteroscedasticity, and $f(Z\gamma)$ describes the functional relationship between these variables and the error variance. Under the null hypothesis:

$$H_0 : \gamma_1 = \gamma_2 = \cdots = \gamma_q = 0 \quad (q \text{ constraints}), \tag{36}$$

the error variance is constant, meaning there is no contribution of Z to heteroscedasticity.
The BP test statistic is expressed as:

$$LM_{BP} = \frac{1}{2\hat{\gamma}_0^2}(\hat{\varepsilon} - \hat{\gamma}_0)' Z (Z'Z)^{-1} Z' (\hat{\varepsilon} - \hat{\gamma}_0), \tag{37}$$

where $\hat{\varepsilon}$ are the residuals from the OLS regression, $\hat{\gamma}_0 = \frac{\hat{\varepsilon}'\hat{\varepsilon}}{n}$ is the OLS estimate of the error variance, and n is the number of observations. In practice, this statistic is simplified to:

$$LM_{BP} = nR^2, \tag{38}$$

where R^2 is the coefficient of determination from the auxiliary regression of $\left(\hat{\varepsilon}^2/\hat{\gamma}_0 - 1\right)$ on Z. The BP test statistic follows a chi-squared distribution with q degrees of freedom: $LM_{BP} \sim \chi_q^2$.

To calculate the BP test statistic, the following steps are followed:

1. Estimate the regression model $Y = X\beta + \varepsilon$ using OLS and obtain the residuals $\hat{\varepsilon}$.
2. Compute the variance estimator $\hat{\gamma}_0 = \frac{\hat{\varepsilon}'\hat{\varepsilon}}{n}$.
3. Perform the auxiliary regression of $\left(\hat{\varepsilon}^2/\hat{\gamma}_0 - 1\right)$ on the explanatory variables in Z.
4. Extract R^2 from the auxiliary regression and calculate $LM_{BP} = nR^2$.

In panel data models, the BP test can also be used to assess the presence of individual effects. In this context, the null hypothesis is:

$$H_0 : \sigma_\mu^2 = 0, \tag{39}$$

indicating no individual-specific variance. If the null holds, the covariance matrix Ω is diagonal, meaning homoscedasticity is present. The BP test for panel data is based on the residuals of the restricted model (typically the fixed effects model), with the test statistic given by:

$$LM_{BP} = \frac{NT}{2(T-1)} \left[\frac{\hat{\varepsilon}'_{wit} \left(I_N \otimes \iota_T \iota'_T\right) \hat{\varepsilon}_{wit}}{\hat{\varepsilon}'_{wit} \hat{\varepsilon}_{wit}} - 1 \right]^2, \tag{40}$$

where $\hat{\varepsilon}_{wit}$ are the residuals from the within estimator, I_N is the identity matrix of size N, and ι_T is a vector of ones of size T.
The BP test statistic for panel data follows a chi-squared distribution with one degree of freedom: χ_1^2. This makes the BP test a computationally simple yet effective tool for detecting heteroscedasticity or testing the significance of individual effects in both cross-sectional and panel data settings.

2.4.2 Hausman Test

Another important procedure for selecting the appropriate panel data model is the Hausman Test (Hausman, 1978). This test evaluates whether to use a fixed effects or random effects estimator. The fixed effects model (also known as the "within estimator") can be costly in terms of model efficiency because it reduces degrees of freedom by excluding time-invariant variables. In contrast, the random effects model is more efficient but relies on the crucial assumption that individual-specific effects are uncorrelated with the regressors. If this assumption does not hold, the random effects model becomes inconsistent. The Hausman test assesses the validity of this assumption to guide the choice of model.
Letting $u = \mu \otimes \iota_T + \varepsilon$, the Hausman test is used to examine the null hypothesis:

$$H_0 : E(X'u) = 0, \tag{41}$$

$$H_1 : E(X'u) \neq 0. \tag{42}$$

The test compares the within (OLS) and GLS estimators under the following conditions:

	H_0	H_1
$\hat{\beta}_{OLS}$	consistent, inefficient	consistent
$\hat{\beta}_{GLS}$	consistent, efficient	inconsistent

The basis of the test is the difference $\hat{q} = \hat{\beta}_{OLS} - \hat{\beta}_{GLS}$. If this difference is statistically insignificant, the random effects model is preferred. However, if $\hat{q}$ is significantly different from zero, the fixed effects (within) estimator is more appropriate.
The test statistic is calculated as:

$$H = \hat{q}'[\text{Var}(\hat{q})]^{-1}\hat{q}, \tag{43}$$

where

$$\text{Var}(\hat{q}) = \text{Var}(\hat{\beta}_{OLS}) + \text{Var}(\hat{\beta}_{GLS}) + 2\text{Cov}(\hat{\beta}_{OLS}, \hat{\beta}_{GLS}). \tag{44}$$

Under H_0, the covariance between the OLS and GLS estimators is zero. To verify this, consider a linear combination of the estimators, defined as:

$$\tilde{\beta} = \hat{\beta}_{GLS} + \lambda\hat{\beta}_{OLS}, \tag{45}$$

where λ is a scalar. Calculating its variance yields:

$$\text{Var}(\tilde{\beta}) = \text{Var}(\hat{\beta}_{GLS}) + \lambda^2\text{Var}(\hat{\beta}_{OLS}) + 2\lambda\text{Cov}(\hat{\beta}_{OLS}, \hat{\beta}_{GLS}). \tag{46}$$

By definition, $\text{Var}(\tilde{\beta}) - \text{Var}(\hat{\beta}_{GLS}) \geq 0$. Therefore, the quadratic expression must also be non-negative:

$$\lambda[\lambda\text{Var}(\hat{\beta}_{OLS}) + 2\text{Cov}(\hat{\beta}_{OLS}, \hat{\beta}_{GLS})] \geq 0. \tag{47}$$

The solutions for this inequality are $\lambda \leq 0$ or $\lambda \geq -\frac{2\text{Cov}(\hat{\beta}_{OLS}, \hat{\beta}_{GLS})}{\text{Var}(\hat{\beta}_{OLS})}$. The condition $\text{Var}(\tilde{\beta}) - \text{Var}(\hat{\beta}_{GLS})$ is guaranteed to be non-negative for any λ if, and only if, the covariance $\text{Cov}(\hat{\beta}_{OLS}, \hat{\beta}_{GLS})$ is zero.
Thus, under H_0, $\hat{q}$ simplifies to:

$$\hat{q} = \text{Var}(\hat{\beta}_{OLS}) + \text{Var}(\hat{\beta}_{GLS}). \tag{48}$$

The test statistic follows a chi-squared distribution: $H \sim \chi^2_k$, where k represents the number of regressors (columns in X).
To provide an example, suppose we have panel data and are considering whether to use a fixed effects or random effects model. After conducting the Hausman test, we calculate the test statistic H and compare it to the critical value of the χ^2 distribution with k degrees of freedom.

- Case 1: Fail to reject H_0 (e.g. H is less than the critical value)
  This indicates that there is no significant correlation between the individual effects and the regressors. In this case, the random effects model is appropriate because it is both consistent and efficient.
- Case 2: Reject H_0 (e.g. H exceeds the critical value) This implies that the individual effects are correlated with the regressors, violating the assumptions of the random effects model. In this scenario, the fixed effects model is preferred because it provides consistent estimates, even if it sacrifices efficiency.

2.4.3 Practical Implementation

In this code we show a practical guide for performing essential model selection and diagnostic tests in panel data econometrics. Using the Grunfeld investment dataset as our working example, we demonstrate a systematic approach that researchers can follow in their own analyses. First, we estimate both Fixed Effects and Random Effects specifications to model investment as a function of market value and capital stock. The core diagnostic element comes next, where we implement the Hausman test—a crucial tool for determining which estimator is more appropriate for your data. This test evaluates whether the individual effects are correlated with the regressors, helping you make an evidence-based choice between the consistent but potentially inefficient Fixed Effects model and the more efficient but potentially inconsistent Random Effects alternative. We also demonstrate how to conduct the Breusch-Pagan test to detect heteroscedasticity in your model's residuals, which can inform your inference strategy and potentially suggest the need for robust

standard errors. By following this structured testing approach, researchers can build more reliable panel data models that respect the underlying data-generating process rather than making arbitrary specification choices. This testing framework provides statistical guidance that can strengthen the methodological foundations of empirical research using panel data.

```
# Import necessary libraries
import numpy as np
import pandas as pd
import statsmodels.api as sm
from linearmodels.panel import PanelOLS, RandomEffects, compare
from statsmodels.stats.diagnostic import het_breuschpagan

# Load Grunfeld dataset (available online)
url = "https://raw.githubusercontent.com/vincentarelbundock/Rdatasets/master/
    csv/plm/Grunfeld.csv"
data = pd.read_csv(url)

# Display the first rows
print(data.head())

# Set multi-index for panel data
data = data.set_index(['firm', 'year'])

# Define the dependent and independent variables
y = data['inv']
X = data[['value', 'capital']]
X = sm.add_constant(X)

# Fixed Effects (Within estimator)
fixed_effects_model = PanelOLS(y, X, entity_effects=True)
fe_results = fixed_effects_model.fit()
print("Fixed Effects Model Results:")
print(fe_results.summary)

# Random Effects estimator
random_effects_model = RandomEffects(y, X)
re_results = random_effects_model.fit()
print("Random Effects Model Results:")
print(re_results.summary)

# Hausman Test for model selection
print("\nHausman Test for Fixed vs Random Effects:")
comparison = compare({'Fixed Effects': fe_results, 'Random Effects':
    re_results})
print(comparison)

# Breusch-Pagan Test for heteroscedasticity using residuals from Random
```

```
    Effects
print("\nBreusch-Pagan Test for Heteroscedasticity:")

# Extract residuals as numpy array
residuals = re_results.resids.values

# Correctly convert exogenous variables to numpy array
exog = re_results.model.exog.dataframe.values

bp_test = het_breuschpagan(residuals, exog)
bp_labels = ['LM Statistic', 'LM-Test p-value', 'F-Statistic', 'F-Test p-value
    ']
print(dict(zip(bp_labels, bp_test)))
```

---

3 Dynamic Panel Models

Dynamic panel models represent an important extension of panel data analysis, allowing for the inclusion of lagged dependent variables among the regressors. This setup is particularly valuable for modeling dynamic processes where current outcomes depend on past values. A key feature of dynamic panels is their ability to distinguish between two types of correlation:

1. **True Correlation:** Autocorrelation of the dependent variable, which arises naturally due to the inclusion of lagged dependent variables.
2. **Spurious Correlation:** Correlation induced by unobserved heterogeneity, which stems from time-invariant individual-specific effects.

To simplify the discussion, consider a dynamic panel model with a single lagged dependent variable. The general equation is:

$$y_{it} = X'_{it}\beta + \phi y_{it-1} + u_{it}, \tag{49}$$

where $u_{it} = \mu_i + \varepsilon_{it}$, μ_i represents unobserved individual effects, and ϕ is the autoregressive parameter. The lagged dependent variable y_{it-1} captures the dynamic nature of the process, while X_{it} is a vector of explanatory variables with corresponding coefficients β.

3.1 Challenges in Estimation: Endogeneity and Bias

The primary challenge in estimating dynamic panel models arises from the correlation between the lagged dependent variable y_{it-1} and the composite error term u_{it}. This correlation renders both ordinary least squares (OLS) and generalized least squares (GLS) estimators inconsistent. The issue can be demonstrated as follows:

$$E(u_{it}y_{it-1}) = E\left[u_{it}\left(X'_{it-1}\beta + \phi y_{it-2} + u_{it-1}\right)\right]. \tag{50}$$

Expanding $u_{it} = \mu_i + \varepsilon_{it}$ and substituting:

$$E(u_{it}y_{it-1}) = E\left[(\mu_i + \varepsilon_{it})\left(X'_{it-1}\beta + \phi y_{it-2} + \mu_i + \varepsilon_{it-1}\right)\right]. \tag{51}$$

The term $E(\mu_i^2) = \sigma_\mu^2$ does not vanish, indicating that y_{it-1} is correlated with u_{it}. Consequently, OLS and GLS estimators are biased and inconsistent in this setting.

3.2 Within Transformation and Bias in Fixed Effects Estimation

The within transformation is a common approach to address unobserved heterogeneity in panel data models by eliminating individual fixed effects (μ_i). This is achieved by subtracting the individual-specific time averages from each observation. Applying this transformation to the dynamic panel model in equation (49) yields:

$$y_{it} - \bar{y}_i = (X_{it} - \bar{X}_i)'\beta + \phi(y_{it-1} - \bar{y}_i) + (\varepsilon_{it} - \bar{\varepsilon}_i), \tag{52}$$

where $\bar{y}_i = \frac{1}{T}\sum_{t=1}^{T} y_{it}$ and similarly for $\bar{X}_i$ and $\bar{\varepsilon}_i$. This transformation removes the fixed effects (μ_i), isolating the time-varying components of the variables.
However, this approach introduces a critical limitation: the lagged dependent variable $(y_{it-1} - \bar{y}_i)$ remains correlated with the transformed error term $(\varepsilon_{it} - \bar{\varepsilon}_i)$ because y_{it-1} is a function of ε_{it-1}. The expected value of the bias is:

$$E[(y_{it-1} - \bar{y}_i)(\varepsilon_{it} - \bar{\varepsilon}_i)] = -\frac{1}{T}\sigma_\varepsilon^2 \neq 0. \tag{53}$$

This bias implies that the within estimator is inconsistent when the number of time periods (T) is finite. As $T \to \infty$, the bias diminishes, and the estimator becomes consistent. Nevertheless, for typical panel data settings with limited T, the within transformation is insufficient for obtaining unbiased estimates of dynamic panel models.

3.3 Anderson-Hsiao Estimator

The Anderson-Hsiao estimator (Anderson and Hsiao, 1981) provides an alternative approach to address the endogeneity of the lagged dependent variable in dynamic panel models. By rewriting equation (49) in first differences, we eliminate the individual fixed effects (μ_i):

$$\Delta y_{it} = \Delta X_{it}'\beta + \phi \Delta y_{it-1} + \Delta \varepsilon_{it}, \tag{54}$$

where Δ denotes first differences, such that $\Delta y_{it} = y_{it} - y_{it-1}$, and $\Delta \varepsilon_{it} = \varepsilon_{it} - \varepsilon_{it-1}$. In this transformation, the fixed effects μ_i drop out because they are time-invariant. However, even after differencing, the lagged dependent variable Δy_{it-1} is correlated with the error term $\Delta \varepsilon_{it}$. This is because:

$$E[(y_{it-1} - y_{it-2})(\varepsilon_{it} - \varepsilon_{it-1})] = E[-y_{it-1}\varepsilon_{it-1}] \neq 0. \tag{55}$$

To resolve this issue, the Anderson-Hsiao estimator employs instrumental variable (IV) techniques. A valid instrument for Δy_{it-1} is y_{it-2}, as it satisfies the following conditions:

- y_{it-2} is correlated with Δy_{it-1} (relevance condition),
- y_{it-2} is uncorrelated with the differenced error term $\Delta \varepsilon_{it}$ (exogeneity condition).

Mathematically, the validity of y_{it-2} as an instrument relies on:

$$E(y_{it-2}\Delta \varepsilon_{it}) = 0. \tag{56}$$

This approach overcomes the endogeneity problem, providing consistent estimates of ϕ and β.

Both the within transformation and the Anderson-Hsiao estimator aim to address unobserved heterogeneity by eliminating individual fixed effects. However, they differ fundamentally in how they deal with the endogeneity of the lagged dependent variable:

- The within transformation removes μ_i but fails to address the correlation between y_{it-1} and ε_{it}, resulting in biased estimates for finite T.
- The Anderson-Hsiao estimator resolves this correlation by using lagged levels of the dependent variable (e.g., y_{it-2}) as instruments for Δy_{it-1}, ensuring consistent estimates even when T is small.

3.4 Arellano-Bond Estimator

The Arellano-Bond estimator (Arellano and Bond, 1991) is a generalized method of moments (GMM) approach designed to estimate dynamic panel data models. It is particularly effective in addressing the endogeneity of the lagged dependent variable and unobserved individual effects, providing consistent estimates even for short time dimensions (T) and a large number of individuals (N).

3.4.1 Pure Autoregressive Model

To simplify the explanation of the Arellano-Bond estimator, consider a pure autoregressive model where exogenous regressors are omitted ($\beta = 0$):

$$y_{it} = \phi y_{it-1} + \mu_i + \varepsilon_{it}. \tag{57}$$

The key assumptions underlying this approach are:

- The time dimension T is fixed,
- The number of individuals $N \to \infty$,
- The idiosyncratic errors ε_{it} are i.i.d. with $E(\varepsilon_{it}) = 0$ and $\text{Var}(\varepsilon_{it}) = \sigma_\varepsilon^2$.

Transforming equation (57) into first differences eliminates the individual effects μ_i:

$$\Delta y_{it} = \phi \Delta y_{it-1} + \Delta \varepsilon_{it} = \phi(y_{it-1} - y_{it-2}) + (\varepsilon_{it} - \varepsilon_{it-1}). \tag{58}$$

Here, the differenced error term $\Delta \varepsilon_{it}$ follows a moving average process of order one ($MA(1)$). The key challenge in this model is that Δy_{it-1} is endogenous, as it is correlated with $\Delta \varepsilon_{it}$ through ε_{it-1}. To resolve this issue, the Arellano-Bond method uses valid instruments derived from lagged levels of the dependent variable.

3.4.2 Instrumental Variable Setup

For each individual, the system of first-differenced equations involves $(T-2)$ equations. Instruments are constructed using lagged levels of the dependent variable, which are uncorrelated with the first-differenced error term $\Delta \varepsilon_{it}$. The instruments are progressively defined as follows:

$$\Delta y_{i3} = \phi \Delta y_{i2} + \Delta \varepsilon_{i3}, \quad \text{instruments: } y_{i1}, \tag{59}$$

$$\Delta y_{i4} = \phi \Delta y_{i3} + \Delta \varepsilon_{i4}, \quad \text{instruments: } y_{i1}, y_{i2}, \tag{60}$$

$$\vdots \tag{61}$$

$$\Delta y_{iT} = \phi \Delta y_{iT-1} + \Delta \varepsilon_{iT}, \quad \text{instruments: } y_{i1}, y_{i2}, \ldots, y_{iT-2}. \tag{62}$$

The lagged levels are valid instruments because they are correlated with Δy_{it-1} and uncorrelated with $\Delta\varepsilon_{it}$.

3.4.3 Variance-Covariance Structure

The variance-covariance matrix of the differenced errors, $\Delta\varepsilon_{it}$, plays a crucial role in the Arellano-Bond estimator. For the i-th individual, this matrix is symmetric and has a tridiagonal structure, given by:

$$V_i = E(\Delta\varepsilon_i \Delta\varepsilon_i') = \sigma_\varepsilon^2 \begin{bmatrix} 2 & -1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & -1 & 2 \end{bmatrix}. \tag{63}$$

For all individuals, the combined variance-covariance matrix is:

$$V = I_N \otimes V_i, \tag{64}$$

where V has dimension $N(T-2) \times N(T-2)$. The instrument matrix for the i-th individual, Z_i, is defined as:

$$Z_i = \begin{bmatrix} y_{i1} & 0 & \dots & 0 & \\ 0 & y_{i1} & y_{i2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & y_{iT-2} & \end{bmatrix}, \tag{65}$$

and the full instrument matrix is:

$$Z = I_N \otimes Z_i. \tag{66}$$

The validity of these instruments depends on the moment condition:

$$E(Z'\Delta\varepsilon) = 0. \tag{67}$$

3.4.4 Estimator Definition

The Arellano-Bond estimator is obtained using a two-step procedure. In the first step, a consistent but inefficient GMM estimator is computed. In the second step, the residuals from the first stage are used to construct a more efficient estimator. The general form of the estimator is:

$$\hat{\phi} = (\Delta Y_{t-1}' Z \Omega^{-1} Z' \Delta Y_{t-1})^{-1} \Delta Y_{t-1}' Z \Omega^{-1} Z' \Delta Y_t, \tag{68}$$

where $\Omega = Z'VZ$ is the estimated variance-covariance matrix of the instruments.

3.4.5 Comparison with the Anderson-Hsiao Estimator

Both the Arellano-Bond and Anderson-Hsiao estimators address the endogeneity of the lagged dependent variable, but they differ in efficiency and implementation:

- **Efficiency:** The Anderson-Hsiao estimator uses only one lag as an instrument, which limits efficiency. In contrast, the Arellano-Bond estimator uses all available valid lags, improving efficiency by exploiting additional moment conditions.
- **Robustness:** The Anderson-Hsiao estimator is simpler and less sensitive to misspecification of the error structure. However, the Arellano-Bond estimator can become inconsistent if the model's assumptions, particularly the lack of serial correlation in errors, are violated.
- **Applicability:** The Anderson-Hsiao estimator is more appropriate for small samples, where the complexity of Arellano-Bond may not be justified. The Arellano-Bond estimator is preferred for large panels with many individuals and short time dimensions.

In summary, while both estimators are consistent, the Arellano-Bond estimator is typically favored in applications requiring efficient use of panel data and complex error structures.

3.4.6 Practical Implementation

In this code, using the Grunfeld investment dataset, we demonstrate the implementation and comparison of different dynamic panel data estimators for modeling firm investment behavior. We first implement the Anderson-Hsiao estimator, which addresses the endogeneity problem in dynamic panels through a two-stage least squares approach with first-differencing to eliminate firm fixed effects. We then showcase two variants of the Arellano-Bond estimator: a basic difference GMM with one lag and an extended version with two lags, both using the pydynpd package. Additionally, we implement the System GMM (Arellano-Bover/Blundell-Bond) estimator, which combines difference and level equations for potentially greater efficiency. For each GMM approach, we examine important diagnostic tests including the Hansen test for overidentifying restrictions and the Arellano-Bond tests for first and second-order autocorrelation in the differenced errors. Finally, we systematically compare these estimators in terms of their coefficient estimates, standard errors, significance levels, and relative efficiency, providing a comprehensive analysis of the strengths and limitations of different dynamic panel approaches when applied to investment modeling.

```
# Dynamic Panel Estimators
import pandas as pd
import numpy as np
import statsmodels.api as sm
from pydynpd import regression
import matplotlib.pyplot as plt

# Load the Grunfeld dataset
url = "https://raw.githubusercontent.com/vincentarelbundock/Rdatasets/master/
    csv/plm/Grunfeld.csv"
data = pd.read_csv(url)
data = data.set_index(['firm', 'year'])

# Display basic information about the dataset
print("Dataset Overview:")
print(data.head())
print("\nDataset contains {} observations across {} firms over {} years".
    format(
     len(data), data.index.get_level_values('firm').nunique(),
```

```
    data.index.get_level_values('year').nunique()))
# The Grunfeld dataset contains 200 observations for 10 firms over 20 years
    (1935-1954)
# It includes investment (inv), market value (value), and capital stock (
    capital) variables
# This balanced panel dataset is frequently used in econometric studies of
    investment behavior

#-------------------------------------------------------------------------------
# Part 1: Anderson-Hsiao Estimator Implementation
#-------------------------------------------------------------------------------
print("\n" + "="*80)
print("ANDERSON-HSIAO ESTIMATOR IMPLEMENTATION")
print("="*80)

# Reset index to manipulate data more easily
panel_data = data.reset_index()

# Create lagged variables for the dependent variable (inv)
panel_data['L1.inv'] = panel_data.groupby('firm')['inv'].shift(1)
panel_data['L2.inv'] = panel_data.groupby('firm')['inv'].shift(2)

# Calculate first differences to eliminate fixed effects
panel_data['D.inv'] = panel_data.groupby('firm')['inv'].diff()
panel_data['D.L1.inv'] = panel_data.groupby('firm')['L1.inv'].diff()
panel_data['D.value'] = panel_data.groupby('firm')['value'].diff()
panel_data['D.capital'] = panel_data.groupby('firm')['capital'].diff()

# Drop rows with missing values due to lagging and differencing
anderson_hsiao_data = panel_data.dropna(subset=['D.inv', 'D.L1.inv', 'L2.inv',
    'D.value', 'D.capital']).copy()

# First stage regression: Regress D.L1.inv on instruments (L2.inv, D.value, D.
    capital)
X_first_stage = sm.add_constant(anderson_hsiao_data[['L2.inv', 'D.value', 'D.
    capital']])
first_stage = sm.OLS(anderson_hsiao_data['D.L1.inv'], X_first_stage).fit()

# Get predicted values from first stage (properly using .loc to avoid
    SettingWithCopyWarning)
anderson_hsiao_data.loc[:, 'D.L1.inv_hat'] = first_stage.predict(X_first_stage
    )

# Second stage regression: Use predicted values in main equation
X_second_stage = sm.add_constant(anderson_hsiao_data[['D.L1.inv_hat', 'D.value
    ', 'D.capital']])
second_stage = sm.OLS(anderson_hsiao_data['D.inv'], X_second_stage).fit()
```

```
# Display results of Anderson-Hsiao estimator
print("\nAnderson-Hsiao First Stage Regression Results:")
print(first_stage.summary().tables[1])

print("\nAnderson-Hsiao Second Stage Regression Results:")
print(second_stage.summary().tables[1])

# Store the Anderson-Hsiao results for comparison
ah_coef = second_stage.params['D.L1.inv_hat']
ah_se = second_stage.bse['D.L1.inv_hat']
ah_pvalue = second_stage.pvalues['D.L1.inv_hat']

print(f"\nAnderson-Hsiao estimate of autoregressive parameter (  ): {ah_coef
    :.4f}")
print(f"Standard error: {ah_se:.4f}")
print(f"p-value: {ah_pvalue:.4f}")
# The Anderson-Hsiao estimator shows a negative autoregressive parameter
    (-0.4431)
# This suggests that higher investment in the previous period leads to lower
    investment in the current period
# However, this effect is not statistically significant (p = 0.4128) and has a
     large standard error (0.5397)
# The first stage shows that the instrument L2.inv is marginally significant (
    p = 0.056)
# Change in value (D.value) is not significant in the first stage, but capital
     (D.capital) is strongly significant
# In the second stage, change in firm value (D.value) has a significant
    positive effect on investment
# Change in capital (D.capital) is marginally significant (p = 0.074) with a
    positive coefficient

#-------------------------------------------------------------------------------
# Part 2: Arellano-Bond Estimator Implementation
#-------------------------------------------------------------------------------
print("\n" + "="*80)
print("ARELLANO-BOND ESTIMATOR IMPLEMENTATION")
print("="*80)

# Reset the index to use with pydynpd (needs firm and year as columns)
reset_data = data.reset_index()

# Basic Arellano-Bond model with one lag (difference GMM)
print("\n1. Basic Arellano-Bond Difference GMM:")
command_str = 'inv L1.inv value capital | gmm(inv, 2:4) gmm(value, 1:2) iv(
    capital) | nolevel'
ab_model = regression.abond(command_str, reset_data, ['firm', 'year'])
```

```
print(ab_model)
# The Arellano-Bond difference GMM shows a strong positive effect of lagged
    investment (0.6116)
# This effect is highly significant (p < 0.0001), contradicting the Anderson-
    Hsiao results
# Both value and capital have significant positive effects on investment
# The model uses 88 instruments for 10 firms, potentially causing instrument
    proliferation issues
# Hansen test p-value is 1.000, which may indicate too many instruments
    relative to groups
# Arellano-Bond AR(1) test is not significant (p = 0.171), which is unusual
    for differenced errors
# The AR(2) test is not significant (p = 0.583), which supports valid moment
    conditions
# Warning: this panel has T > N (20 years > 10 firms), which is suboptimal for
     GMM estimators

# Extract key coefficients for comparison
ab_coef = ab_model.models[0].regression_table.iloc[0]['coefficient']
ab_se = ab_model.models[0].regression_table.iloc[0]['std_err']
ab_pvalue = ab_model.models[0].regression_table.iloc[0]['p_value']

# Model with two lags (still difference GMM)
print("\n2. Arellano-Bond with two lags:")
command_str_2lags = 'inv L1.inv L2.inv value capital | gmm(inv, 2:4) gmm(value
    , 1:2) iv(capital) | nolevel'
ab_model_2lags = regression.abond(command_str_2lags, reset_data, ['firm', '
    year'])
print(ab_model_2lags)
# The two-lag Arellano-Bond model confirms the strong positive first lag
    effect (0.7070, p < 0.0001)
# The second lag shows a negative effect (-0.2403) but is not statistically
    significant (p = 0.2528)
# Value and capital remain significant with positive coefficients
# The Hansen test p-value of 1.000 continues to suggest instrument
    proliferation issues
# AR tests show similar patterns to the one-lag model (AR(1) p = 0.128, AR(2)
    p = 0.442)
# The non-significant second lag suggests the one-lag specification may be
    sufficient

# System GMM model
print("\n3. System GMM (Arellano-Bover/Blundell-Bond):")
command_str_sys = 'inv L1.inv value capital | gmm(inv, 2:4) gmm(value, 1:2) iv
    (capital)'
sys_model = regression.abond(command_str_sys, reset_data, ['firm', 'year'])
print(sys_model)
```

```
# System GMM shows an even stronger autoregressive effect (0.8940, p < 0.0001)
# This suggests high persistence in firm investment behavior
# The effect of firm value is much smaller (0.0265) though still significant (
    p = 0.0317)
# Capital becomes non-significant in this specification (p = 0.3485)
# System GMM adds a constant term (-17.22), but it's not statistically
    significant
# The number of instruments increases to 126, exacerbating instrument
    proliferation concerns
# The much stronger autoregressive coefficient suggests that difference GMM
    may underestimate persistence
# The AR(2) test (p = 0.145) remains non-significant, supporting the validity
    of the model

# Extract system GMM coefficient
sys_coef = sys_model.models[0].regression_table.iloc[0]['coefficient']
sys_se = sys_model.models[0].regression_table.iloc[0]['std_err']
sys_pvalue = sys_model.models[0].regression_table.iloc[0]['p_value']

#-----------------------------------------------------------------------------
# Part 3: Diagnostic Tests
#-----------------------------------------------------------------------------
print("\n" + "="*80)
print("DIAGNOSTIC TESTS")
print("="*80)

# Based on the package structure, we need to use the correct attributes
# For pydynpd models, these tests may be accessed differently

# It appears the actual attribute names are different
# Let's try to print what attributes are available in the model object
print("\nAvailable attributes and methods in the model object:")
model_attrs = [attr for attr in dir(ab_model.models[0]) if not attr.startswith
    ('_')]
print(model_attrs)
# The model object has many attributes but does not directly expose test
    statistics as attributes
# Instead, we need to extract them from the printed output or access them via
    internal methods
# Key attributes include 'hansen', 'regression_table', and various methods for
     model information

# Since we can't programmatically extract the test results easily,
# let's manually report them based on the printed output
print("\nArellano-Bond Diagnostic Tests (from printed output):")
print("AR(1) test in first differences: z = -1.37, p-value = 0.171")
print("AR(2) test in first differences: z = -0.55, p-value = 0.583")
```

```
print("Hansen test of overidentifying restrictions: chi2(85) = 8.336, p-value
    = 1.000")
# The Hansen test p-value of 1.000 is suspicious and likely indicates
    instrument proliferation
# When there are too many instruments relative to groups, the Hansen test
    loses power
# The AR(1) test is not significant at conventional levels (p = 0.171), which
    is unusual
# Typically, we expect significant negative first-order autocorrelation in
    differenced errors
# The AR(2) test is not significant (p = 0.583), which is good for instrument
    validity
# This confirms that lags of order 2 and higher are valid instruments

# Alternatively, if we need the test values for comparisons, we can
# manually set them based on the printed output
ar1_z = -1.37
ar1_pvalue = 0.171
ar2_z = -0.55
ar2_pvalue = 0.583
hansen_chi2 = 8.336
hansen_df = 85
hansen_pvalue = 1.000

#-------------------------------------------------------------------------------
# Part 4: Comparison of Methods
#-------------------------------------------------------------------------------
print("\n" + "="*80)
print("COMPARISON OF METHODS")
print("="*80)

# Create a comparison table
comparison_df = pd.DataFrame({
    'Estimator': ['Anderson-Hsiao', 'Arellano-Bond', 'System GMM'],
    'Coefficient (  )': [ah_coef, 0.6116201, 0.8939961],  # Using values from
        printed output
    'Standard Error': [ah_se, 0.1223293, 0.0929766],      # Using values from
        printed output
    'p-value': [ah_pvalue, 0.0000006, 0.0000000]          # Using values from
        printed output
})

print("\nComparison of autoregressive parameter estimates (  ):")
print(comparison_df)

# Calculate relative efficiency using the values from printed output
relative_efficiency_ab = (ah_se**2) / (0.1223293**2)
```

```
relative_efficiency_sys = (ah_se**2) / (0.0929766**2)

print(f"\nRelative efficiency (Arellano-Bond vs. Anderson-Hsiao): {
    relative_efficiency_ab:.2f}")
print(f"Relative efficiency (System GMM vs. Anderson-Hsiao): {
    relative_efficiency_sys:.2f}")
# The comparison shows dramatic differences in estimated investment
    persistence:
# - Anderson-Hsiao: negative coefficient (-0.44) that is not statistically
    significant
# - Arellano-Bond: moderate positive coefficient (0.61) that is highly
    significant
# - System GMM: strong positive coefficient (0.89) that is highly significant
# The GMM estimators are much more efficient than Anderson-Hsiao:
# - Arellano-Bond is 19.47 times more efficient
# - System GMM is 33.70 times more efficient
# This efficiency comes from using more instruments and optimal weighting
# The stark difference in estimates highlights how important estimation method
    can be
# However, given the potential instrument proliferation issues, these
    efficiency gains
# may come at the cost of potential bias, especially in the System GMM model
```

---

4 Difference-in-Differences in Panel Models

The Difference-in-Differences (DiD) estimator is one of the most widely used methods in applied econometrics for identifying causal effects, particularly in the evaluation of policy interventions and treatment effects. Its popularity stems from its intuitive appeal and ability to control for both observed and unobserved time-invariant confounders. In panel data settings, the DiD framework leverages repeated observations on individuals or units over time, using a natural experiment design where some units (the treatment group) are exposed to a treatment or intervention, while others (the control group) remain unaffected. The central idea is to estimate the causal effect of the treatment by comparing the pre- and post-treatment changes in outcomes between the treatment and control groups (Card and Krueger, 1994; Angrist and Pischke, 2008).

4.1 The Theoretical Setup

Consider a panel dataset with N units (individuals, firms, regions, etc.) observed over T time periods. The outcome variable y_{it} is measured for each unit i in time period t. A subset of units is exposed to a binary treatment D_{it} starting at a specific time t^*, while the remaining units serve as the control group. A general specification for the outcome variable is given by:

$$y_{it} = \alpha_i + \lambda_t + \delta D_{it} + X_{it}'\beta + \varepsilon_{it}, \tag{69}$$

where:

- α_i captures individual-specific fixed effects (time-invariant heterogeneity such as innate productivity or geographical characteristics),

- λ_t represents time-specific effects (common shocks like macroeconomic conditions),
- D_{it} is a binary variable equal to 1 if unit i is treated at time t, and 0 otherwise,
- X_{it} is a vector of time-varying covariates with corresponding coefficients β,
- ε_{it} is the error term.

The key parameter of interest is δ, which represents the average treatment effect on the treated (ATT). By differencing out α_i and comparing pre- and post-treatment changes in y_{it} across groups, the DiD estimator isolates δ under the assumption that the treatment and control groups would have followed parallel trends in the absence of treatment.

4.2 Estimation and Identification

To estimate δ, fixed effects (FE) methods are typically employed, as they effectively remove α_i from the model. Taking first differences transforms (69) into:

$$\Delta y_{it} = \lambda + \delta \Delta D_{it} + \Delta X_{it}' \beta + \Delta \varepsilon_{it}, \tag{70}$$

where Δ denotes differences across consecutive time periods (e.g., $\Delta y_{it} = y_{it} - y_{it-1}$). This approach eliminates α_i, thus controlling for time-invariant unobserved heterogeneity.
The key identifying assumption in the DiD framework is the **parallel trends assumption**. Formally, this requires that in the absence of treatment, the average change in y_{it} would have been the same for the treatment and control groups. Mathematically, this assumption is:

$$E[\Delta y_{it} | D_{it} = 1] - E[\Delta y_{it} | D_{it} = 0] = 0, \quad \text{for } t < t^*. \tag{71}$$

If this assumption holds, δ can be interpreted as the causal effect of the treatment. However, violations of the parallel trends assumption, as highlighted by Roth et al. (2022), can lead to biased estimates. Thus, careful diagnostic checks and robustness tests are essential in empirical applications.

4.2.1 Inference and Robustness

Inference in DiD models requires accounting for potential serial correlation and heteroskedasticity in panel data. Bertrand et al. (2004) demonstrate that ignoring serial correlation can lead to severely underestimated standard errors, resulting in over-rejection of the null hypothesis. They recommend clustering standard errors at the unit level to ensure valid inference. This adjustment corrects for within-unit correlations over time, which are pervasive in most panel datasets.

4.3 Extensions of the Basic DiD Model

The basic DiD framework can be extended in several directions to accommodate more complex empirical settings. These extensions have been critical in broadening the applicability of DiD to modern econometric problems.

Event Study Analysis Event study analysis allows researchers to investigate the dynamic effects of treatment over time. This approach provides insights into pre-treatment trends (parallel trends testing) and post-treatment heterogeneity in treatment effects. Jacobson et al. (1993) use event studies to analyze labor market outcomes following job displacement, illustrating the utility of this method in panel data.

Heterogeneous Treatment Effects The treatment effect δ may vary across individuals or over time. Semiparametric methods, such as those developed by Abadie (2005), allow for heterogeneity in treatment effects while maintaining flexibility in the functional form of the model. This extension is particularly useful when the treatment interacts with observed covariates or unobserved factors.

Multiple Treatment Periods and Staggered Adoption In many applications, treatment does not occur simultaneously for all treated units. For instance, in policy evaluation, different regions or firms may adopt the policy at different times. Callaway and Sant'Anna (2021) extend the DiD framework to accommodate staggered treatment adoption, providing consistent estimators for ATT in such settings.

4.3.1 Practical Implementation

In this practical implementation, we provide examples of DiD analysis using simulated data. The code illustrates how to generate treatment and control groups, and visually examine the differences in outcomes before and after an intervention. Through a series of progressively complex examples, we demonstrate the use of ordinary least squares regression to estimate causal effects while accounting for key assumptions such as parallel trends. Additionally, the implementation extends to include time-varying covariates, multiple time periods, synthetic control methods, propensity score matching, event study designs, and heterogeneous treatment effects. This comprehensive framework serves as a practical guide for applying DiD techniques in empirical studies.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.formula.api import ols
from sklearn.linear_model import LogisticRegression
from scipy.stats import ttest_ind
from scipy.optimize import minimize

##############################################################################
# SECTION 1: INTRODUCTION TO DIFFERENCE-IN-DIFFERENCES (DiD)
##############################################################################
# DiD is a statistical method to estimate causal effects by comparing
# changes in outcomes over time between a treatment and control group.
# This technique is particularly useful when randomized experiments
# are not feasible or ethical.
##############################################################################


# Generate sample data
np.random.seed(42)
time = np.array([0, 1])
control = np.array([50, 52]) + np.random.normal(0, 2, 2)
treatment = np.array([50, 58]) + np.random.normal(0, 2, 2)

# Plot DiD
```

```
plt.figure(figsize=(10, 6))
plt.plot(time, control, 'b-', label='Control Group')
plt.plot(time, treatment, 'r-', label='Treatment Group')
plt.xlabel('Time')
plt.ylabel('Outcome')
plt.title('Difference-in-Differences Illustration')
plt.legend()
plt.xticks([0, 1], ['Before', 'After'])
plt.show()

##############################################################################
# SECTION 2: KEY ASSUMPTIONS OF DiD
##############################################################################
# The DiD method relies on several key assumptions:
# 1. Parallel trends assumption: in the absence of treatment, the
#    difference between treatment and control groups would remain constant
# 2. Stable composition of groups
# 3. No spillover effects between groups
##############################################################################

# Generate data with parallel trends
np.random.seed(42)
time = np.arange(5)
control = 50 + 2 * time + np.random.normal(0, 1, 5)
treatment = 48 + 2 * time + np.random.normal(0, 1, 5)
treatment_effect = np.array([0, 0, 5, 5, 5])

plt.figure(figsize=(10, 6))
plt.plot(time, control, 'b-', label='Control Group')
plt.plot(time, treatment, 'r--', label='Treatment Group (Counterfactual)')
plt.plot(time, treatment + treatment_effect, 'r-', label='Treatment Group (
    Observed)')
plt.axvline(x=2, color='gray', linestyle='--', label='Intervention')
plt.xlabel('Time')
plt.ylabel('Outcome')
plt.title('Parallel Trends Assumption')
plt.legend()
plt.show()

##############################################################################
# SECTION 3: SETTING UP THE DiD MODEL
##############################################################################
# To set up a DiD model, we need to organize data into treatment and
# control groups, as well as pre- and post-intervention periods using
# a structured DataFrame.
##############################################################################
```

```
# Generate sample data
np.random.seed(42)
n = 1000
data = pd.DataFrame({
    'id': range(n),
    'treatment': np.random.choice([0, 1], n),
    'time': np.random.choice([0, 1], n),
    'outcome': np.random.normal(50, 10, n)
})

# Add treatment effect
data.loc[(data['treatment'] == 1) & (data['time'] == 1), 'outcome'] += 5

print(data.head(10))
print("\nData summary:")
print(data.groupby(['treatment', 'time'])['outcome'].mean())

##############################################################################
# SECTION 4: IMPLEMENTING DiD USING ORDINARY LEAST SQUARES (OLS)
##############################################################################
# We can implement the DiD model using OLS regression to estimate the
# treatment effect while controlling for group-specific and time-specific
# effects. The interaction term coefficient represents the DiD estimate.
##############################################################################

# Prepare the data for regression
data['treatment_time'] = data['treatment'] * data['time']
X = sm.add_constant(data[['treatment', 'time', 'treatment_time']])
y = data['outcome']

# Fit the OLS model
model = sm.OLS(y, X).fit()

# Print the results
print(model.summary())

# Extract the DiD estimator (coefficient of treatment_time)
did_estimate = model.params['treatment_time']
print(f"\nEstimated treatment effect: {did_estimate:.4f}")


##############################################################################
# SECTION 5: HANDLING TIME-VARYING COVARIATES
##############################################################################
# In practice, we often need to control for time-varying covariates that
# might affect the outcome. We can incorporate these into our DiD model
# to improve the accuracy of estimates.
```

```
##############################################################################

# Generate sample data with a time-varying covariate
np.random.seed(42)
n = 1000
data = pd.DataFrame({
    'id': range(n),
    'treatment': np.random.choice([0, 1], n),
    'time': np.random.choice([0, 1], n),
    'covariate': np.random.normal(0, 1, n),
    'outcome': np.random.normal(50, 10, n)
})

# Add treatment effect and covariate effect
data.loc[(data['treatment'] == 1) & (data['time'] == 1), 'outcome'] += 5
data['outcome'] += 2 * data['covariate']

# Prepare the data for regression
data['treatment_time'] = data['treatment'] * data['time']
X = sm.add_constant(data[['treatment', 'time', 'treatment_time', 'covariate'
    ]])
y = data['outcome']

# Fit the OLS model
model = sm.OLS(y, X).fit()

# Print the results
print(model.summary())

# Extract the DiD estimator (coefficient of treatment_time)
did_estimate = model.params['treatment_time']
print(f"\nEstimated treatment effect: {did_estimate:.4f}")


##############################################################################
# SECTION 6: DiD WITH MULTIPLE TIME PERIODS
##############################################################################
# DiD can be extended to settings with multiple time periods, allowing for
# more robust estimation of treatment effects over time.
##############################################################################

# Generate sample data with multiple time periods
np.random.seed(42)
n = 1000
periods = 5
data = pd.DataFrame({
    'id': np.repeat(range(n), periods),
```

```
    'treatment': np.repeat(np.random.choice([0, 1], n), periods),
    'time': np.tile(range(periods), n),
    'outcome': np.random.normal(50, 10, n * periods)
})

# Add treatment effect starting from period 3
data.loc[(data['treatment'] == 1) & (data['time'] >= 3), 'outcome'] += 5

# Create dummy variables for each time period
for t in range(1, periods):
    data[f'time_{t}'] = (data['time'] == t).astype(int)

# Create interaction terms
for t in range(1, periods):
    data[f'treat_time_{t}'] = data['treatment'] * data[f'time_{t}']

# Prepare the data for regression
X = sm.add_constant(data[['treatment'] + [f'time_{t}' for t in range(1,
    periods)] +
                         [f'treat_time_{t}' for t in range(1, periods)]])
y = data['outcome']

# Fit the OLS model
model = sm.OLS(y, X).fit()

# Print the results
print(model.summary())

# Plot the treatment effects over time
effects = [model.params[f'treat_time_{t}'] for t in range(1, periods)]
plt.figure(figsize=(10, 6))
plt.plot(range(1, periods), effects, 'bo-')
plt.xlabel('Time Period')
plt.ylabel('Treatment Effect')
plt.title('Treatment Effects Over Time')
plt.axhline(y=0, color='r', linestyle='--')
plt.show()

##############################################################################
# SECTION 7: SYNTHETIC CONTROL METHOD
##############################################################################
# The Synthetic Control Method is an extension of DiD that creates a
# synthetic control unit by weighting control units to match the
# pre-treatment characteristics of the treated unit.
##############################################################################

# Generate sample data
```

```
np.random.seed(42)
n_units = 20
n_periods = 10
data_synth = pd.DataFrame({
    'unit': np.repeat(range(n_units), n_periods),
    'time': np.tile(range(n_periods), n_units),
    'outcome': np.random.normal(50, 10, n_units * n_periods)
})

# Add treatment effect for unit 0 in the last 5 periods
data_synth.loc[(data_synth['unit'] == 0) & (data_synth['time'] >= 5), 'outcome
    '] += 10

# Separate treated and control units
treated = data_synth[data_synth['unit'] == 0]
control = data_synth[data_synth['unit'] != 0]

# Prepare pre-treatment data
treated_pre = treated[treated['time'] < 5].set_index('time')['outcome']
control_units = sorted(control['unit'].unique())
control_pre = control[control['time'] < 5].pivot(index='time', columns='unit',
    values='outcome')

# Define the objective function to minimize
def objective(weights, control_pivot, treated_outcome):
    synthetic = (control_pivot * weights).sum(axis=1)
    return np.sum((treated_outcome - synthetic)**2)

# Optimize weights
result = minimize(
    objective,
    x0=np.ones(len(control_units)) / len(control_units),
    args=(control_pre, treated_pre),
    method='SLSQP',
    constraints={'type': 'eq', 'fun': lambda x: np.sum(x) - 1},
    bounds=[(0, 1) for _ in range(len(control_units))]
)

# Calculate synthetic control for all periods
control_all = control.pivot(index='time', columns='unit', values='outcome')
synthetic = (control_all * result.x).sum(axis=1)

# Plot results
plt.figure(figsize=(10, 6))
plt.plot(treated.set_index('time')['outcome'], label='Treated Unit')
plt.plot(synthetic, label='Synthetic Control')
plt.axvline(x=4.5, color='r', linestyle='--', label='Treatment Start')
```

```
plt.xlabel('Time')
plt.ylabel('Outcome')
plt.title('Synthetic Control Method')
plt.legend()
plt.show()

##############################################################################
# SECTION 8: DiD WITH PROPENSITY SCORE MATCHING
##############################################################################
# Combining DiD with propensity score matching can help address potential
# selection bias. This approach first matches treated and control units
# based on propensity to receive treatment, then applies DiD.
##############################################################################

# Generate sample data
np.random.seed(42)
n = 1000
data = pd.DataFrame({
    'age': np.random.normal(40, 10, n),
    'income': np.random.normal(50000, 10000, n),
    'education': np.random.choice(['low', 'medium', 'high'], n),
    'treatment': np.random.choice([0, 1], n),
    'time': np.random.choice([0, 1], n),
    'outcome': np.random.normal(50, 10, n)
})

# Add treatment effect
data.loc[(data['treatment'] == 1) & (data['time'] == 1), 'outcome'] += 5

# Estimate propensity scores
X = pd.get_dummies(data[['age', 'income', 'education']], drop_first=True)
y = data['treatment']
ps_model = LogisticRegression()
ps_model.fit(X, y)
data['propensity_score'] = ps_model.predict_proba(X)[:, 1]

# Perform matching (simplified nearest neighbor matching)
data['matched'] = False
for treated in data[data['treatment'] == 1].index:
    control = data[(data['treatment'] == 0) & (~data['matched'])]['
        propensity_score'].sub(data.loc[treated, 'propensity_score']).abs().
        idxmin()
    data.loc[[treated, control], 'matched'] = True

# Perform DiD on matched sample
matched_data = data[data['matched']]
did_model = sm.OLS.from_formula('outcome ~ treatment + time + treatment:time',
```

```
    data=matched_data).fit()

print(did_model.summary())

##############################################################################
# SECTION 9: EVENT STUDY DESIGN
##############################################################################
# An event study design extends the DiD framework by examining treatment
# effects over multiple periods before and after intervention. This helps
# visualize pre-treatment trends and dynamic effects.
##############################################################################

# Generate sample data
np.random.seed(42)
n_units = 100
n_periods = 10
treatment_start = 5

data_event = pd.DataFrame({
    'unit': np.repeat(range(n_units), n_periods),
    'time': np.tile(range(n_periods), n_units),
    'treatment': np.repeat(np.random.choice([0, 1], n_units), n_periods),
    'outcome': np.random.normal(50, 10, n_units * n_periods)
})

# Add treatment effect
data_event.loc[(data_event['treatment'] == 1) & (data_event['time'] >=
    treatment_start), 'outcome'] += 5

# Create relative time variable
data_event['rel_time'] = data_event['time'] - treatment_start
data_event['post'] = (data_event['rel_time'] >= 0).astype(int)

# Create dummy variables with proper naming convention for statsmodels
# Use "pre_X" for negative periods and "post_X" for positive periods
for t in range(-treatment_start + 1, n_periods - treatment_start):
    if t != -1:  # Omit -1 as the reference category
        if t < 0:
            col_name = f'pre_{abs(t)}'
        else:
            col_name = f'post_{t}'
        data_event[col_name] = (data_event['rel_time'] == t).astype(int)

# Construct formula with proper variable names
pre_periods = [f'pre_{abs(t)}' for t in range(-treatment_start + 1, 0) if t !=
    -1]
post_periods = [f'post_{t}' for t in range(0, n_periods - treatment_start)]
```

```
formula = 'outcome ~ ' + ' + '.join(pre_periods + post_periods)

# Estimate event study model
model = sm.OLS.from_formula(formula, data=data_event[data_event['treatment']
    == 1]).fit()

# Plot event study results
coef = model.params[1:]
ci = model.conf_int().iloc[1:]

# Create proper x-axis values for plotting
x_values = [t for t in range(-treatment_start + 1, n_periods - treatment_start
    ) if t != -1]
plt.figure(figsize=(12, 6))
plt.plot(x_values, coef, marker='o')
plt.fill_between(x_values, ci[0], ci[1], alpha=0.2)
plt.axvline(x=0, color='r', linestyle='--')
plt.axhline(y=0, color='k', linestyle='-')
plt.xlabel('Relative Time')
plt.ylabel('Treatment Effect')
plt.title('Event Study Results')
plt.show()
##############################################################################
# SECTION 10: HETEROGENEOUS TREATMENT EFFECTS
##############################################################################
# Exploring heterogeneous treatment effects allows understanding how
# the impact of an intervention varies across different subgroups or
# characteristics of the population.
##############################################################################

# Generate sample data
np.random.seed(42)
n = 1000
data = pd.DataFrame({
    'age': np.random.uniform(20, 60, n),
    'treatment': np.random.choice([0, 1], n),
    'time': np.random.choice([0, 1], n),
    'outcome': np.random.normal(50, 10, n)
})

# Add heterogeneous treatment effect
data['outcome'] += 5 * data['treatment'] * data['time']  # Base effect
data['outcome'] += 0.1 * data['age'] * data['treatment'] * data['time']  # Age
    interaction

# Estimate heterogeneous treatment effects
formula = 'outcome ~ treatment + time + treatment:time + age + age:treatment:
```

```
    time'
model = sm.OLS.from_formula(formula, data=data).fit()

print(model.summary())

# Calculate treatment effect at different ages
ages = np.linspace(20, 60, 5)
effects = model.params['treatment:time'] + model.params['age:treatment:time']
    * ages

# Plot heterogeneous treatment effects
plt.figure(figsize=(10, 6))
plt.plot(ages, effects, marker='o')
plt.xlabel('Age')
plt.ylabel('Treatment Effect')
plt.title('Heterogeneous Treatment Effects by Age')
plt.show()


##############################################################################
# SECTION 11: ROBUSTNESS CHECKS AND SENSITIVITY ANALYSIS
##############################################################################
# Conducting robustness checks and sensitivity analyses is crucial to
# ensure the validity and reliability of DiD estimates. These techniques
# help assess stability of results under different assumptions.
##############################################################################

# Generate sample data
np.random.seed(42)
n = 1000
data = pd.DataFrame({
    'treatment': np.random.choice([0, 1], n),
    'time': np.random.choice([0, 1], n),
    'outcome': np.random.normal(50, 10, n),
    'covariate': np.random.normal(0, 1, n)
})

# Add treatment effect
data.loc[(data['treatment'] == 1) & (data['time'] == 1), 'outcome'] += 5

# Base DiD model
base_model = sm.OLS.from_formula('outcome ~ treatment + time + treatment:time'
    , data=data).fit()

# DiD model with covariate
covariate_model = sm.OLS.from_formula('outcome ~ treatment + time + treatment:
    time + covariate', data=data).fit()
```

```
# Placebo test (fake treatment)
data['fake_treatment'] = np.random.choice([0, 1], n)
placebo_model = sm.OLS.from_formula('outcome ~ fake_treatment + time +
    fake_treatment:time', data=data).fit()

# Print results
print("Base DiD Model:")
print(base_model.summary().tables[1])

print("\nDiD Model with Covariate:")
print(covariate_model.summary().tables[1])

print("\nPlacebo Test:")
print(placebo_model.summary().tables[1])
```

4.4 Limitations of DiD in Panel Models

While DiD is a robust and intuitive method for causal inference, it is not without limitations:

- **Violation of Parallel Trends:** If the treatment and control groups exhibit different trends before treatment, the DiD estimator is biased. Recent advances, such as placebo tests and graphical diagnostics (Roth et al., 2022), aim to address this issue.
- **Time-Varying Confounders:** If there are confounders that vary over time and correlate with treatment, the parallel trends assumption is violated, leading to bias in δ.
- **Attrition and Sample Composition:** In panel data, attrition (e.g., individuals dropping out of the study) can induce bias if it is correlated with treatment or outcome variables.

5 Conclusions

This note provides a structured overview of key methods in panel data econometrics, focusing on techniques to address challenges like unobserved heterogeneity and endogeneity. Starting with foundational concepts like fixed and random effects models, it introduces advanced approaches for dynamic panels, including the Anderson-Hsiao and Arellano-Bond estimators, which are designed to handle endogeneity caused by lagged dependent variables.

Additionally, the note covers Difference-in-Differences methods, highlighting their use in causal inference and policy evaluation. These methods leverage both cross-sectional and temporal variations in panel data, offering flexibility in analyzing treatment effects.

By combining theoretical explanations with practical insights, this note serves as a guide to understanding and applying panel data methods effectively, providing a foundation for tackling real-world empirical research challenges.

References

Abadie, A. (2005). Semiparametric difference-in-differences estimators. *Review of Economic Studies*, 72(1):1–19.

Anderson, T. W. and Hsiao, C. (1981). Estimation of dynamic models with error components. *Journal of the American Statistical Association*, 76(375):598–606.

Angrist, J. D. and Pischke, J.-S. (2008). *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton University Press.

Arellano, M. (2003). *Panel Data Econometrics*. Oxford University Press.

Arellano, M. and Bond, S. (1991). Some tests of specification for panel data: Monte carlo evidence and an application to employment equations. *The Review of Economic Studies*, 58(2):277–297.

Baltagi, B. H. (2008). *Econometric Analysis of Panel Data*. John Wiley & Sons.

Bertrand, M., Duflo, E., and Mullainathan, S. (2004). How much should we trust differences-in-differences estimates? *Quarterly Journal of Economics*, 119(1):249–275.

Breusch, T. S. and Pagan, A. R. (1980). The lagrange multiplier test and its applications to model specification in econometrics. *The Review of Economic Studies*, 47(1):239–253.

Callaway, B. and Sant'Anna, P. H. C. (2021). Difference-in-differences with multiple time periods. *Journal of Econometrics*, 225(2):200–230.

Card, D. and Krueger, A. B. (1994). Minimum wages and employment: A case study of the fast food industry in new jersey and pennsylvania. *American Economic Review*, 84(4):772–793.

Hausman, J. A. (1978). Specification tests in econometrics. *Econometrica*, 46(6):1251–1271.

Hsiao, C. (2014). *Analysis of Panel Data*. Cambridge University Press.

Jacobson, L. S., LaLonde, R. J., and Sullivan, D. G. (1993). Earnings losses of displaced workers. *American Economic Review*, 83(4):685–709.

Palomba, G. (2008). Panel data. *Dipartimento di Economia Politica, Univpm*.

Roth, J., Sant'Anna, P. H. C., Bilinski, A., and Poe, J. (2022). What's trending in difference-in-differences? a synthesis of the recent econometrics literature. *Quarterly Journal of Economics*, 137(1):387–433.

Wooldridge, J. M. (2010). *Econometric Analysis of Cross Section and Panel Data*. MIT Press.