Difference in Differences

Workshop 08 Open in Colab

Aims:

This workshop builds on last week's material, replicating analysis in published academic research on the relationship between minimum wages and unemployment.

As always we'll start by importing the libraries I need

```
#!pip install linearmodels
import pandas as pd
import seaborn as sns
import numpy as np
import plotly
import plotly.express as px
import warnings
from statsmodels.formula.api import ols
from statsmodels.iolib.summary2 import summary_col
import matplotlib.pyplot as plt

warnings.filterwarnings('ignore')
sns.set(font_scale=1.5)
sns.set_style("white")
plt.rcParams['figure.figsize'] = (12, 8)
```

Panel Regression

Surveys indicate that "jobs" are consistently one of the most important issues among voters in U.S. presidential elections, and that Republicans are typically perceived as better in handling the economy than Democrats. An article in NBC claims that "analysis of unemployment and voting data found that the president's share of the vote held steady or increased in each of the 20 counties with the highest rise in unemployment from September 2019 to September 2020. And his vote share improved by 1 percentage point or more in 70 of the 100 hardest-hit counties." Let's look into this.

Data Collection

There are only 50 states in the U.S. but there are over 3000 counties-- this allows us to increase our sample size and perform a more fine-grained analysis. This is particularly important if we're interested in investigating the relationship between

unemployment and voting behaviour, because of the urban-rural divide. For example, with in the state of New York there are probably vast differences in social and economic factors relevant to voting behaviour between Manhattan and very rural areas; this variation is lost when we look at aggregate state-level resutls, but visible when we look at the county-level. As such, in addition to the datasets we've just imported, we're going to be downloading county-level unemployment data straight from the BLS using the loop below.

```
1 counties=pd.read_csv('https://storage.googleapis.com/qm2/wk10/county_labor.cs
2 counties.head()
```

	state	county	year	unemployment	population	county_fips
0	1	1	2008	5.3	24.687	01001
1	1	3	2008	4.8	83.205	01003
2	1	5	2008	9.1	10.175	01005
3	1	7	2008	6.0	8.751	01007
4	1	9	2008	4.8	26.693	01009

Next steps: Generate code with `counties` New interactive sheet

Part of the cleaning process in the cell above involves the creation of a column called "county_fips"-- this stands for Federal Information Processing System. This is a code that uniquely identifies states and counties in the U.S. A two digit FIPS code identifies states (e.g. 01: Alabama, 02: Alaska, etc.) and a five digit fips code identifies counties (e.g. 010001: Atauga County, Alabama; 02068: Denali Borough, Alaska). Notice, the first two digits of the five-digit county FIPS code indicates the state. Boring, yes, but these codes are imperative in allowing us to join county- and state- level datasets from different sources quicky and easily. Imagine what a nightmare it would be to try to join them using the names of the counties, having to deal with capitalizations, punctuation, etc. Yikes.

Maps

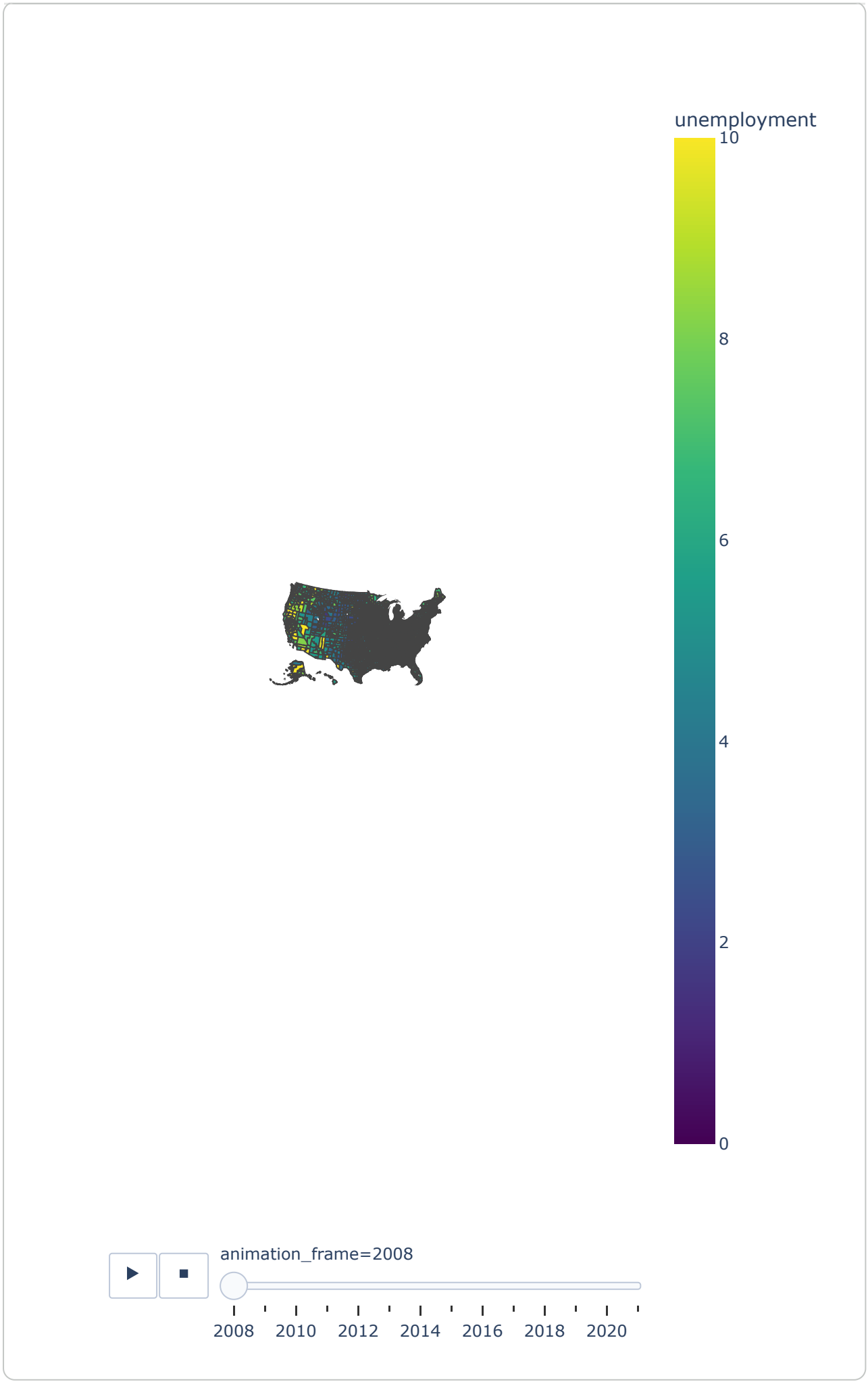
Great-- we've now got clean, county-level unemployment and population data spanning from 1990-2022 on an annual basis. Lets make a map to explore the spatial distribution of unemployment across time in the U.S. In order to do that, we're going to need a spatial file that tells us the shapes of the counties; I've imported it as a variable called `county_polygons`. We're then going to create an map using the Plotly library, which is great for making pretty, interactive maps and plots. It will have a

slider on the bottom that lets us view unemployment in different years. It's doing quite a bit under the hood so it will take some time to plot. Be patient.

```
import json
!mkdir data
!mkdir data/wk10/
!curl https://storage.googleapis.com/qm2/wk10/geojson-counties-fips.json -o d

county_polygons = json.load(open("data/wk10/geojson-counties-fips.json"))
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
                                 Dload  Upload   Total   Spent    Left
100 3141k  100 3141k    0     0  2690k      0  0:00:01  0:00:01 --:--:-
```

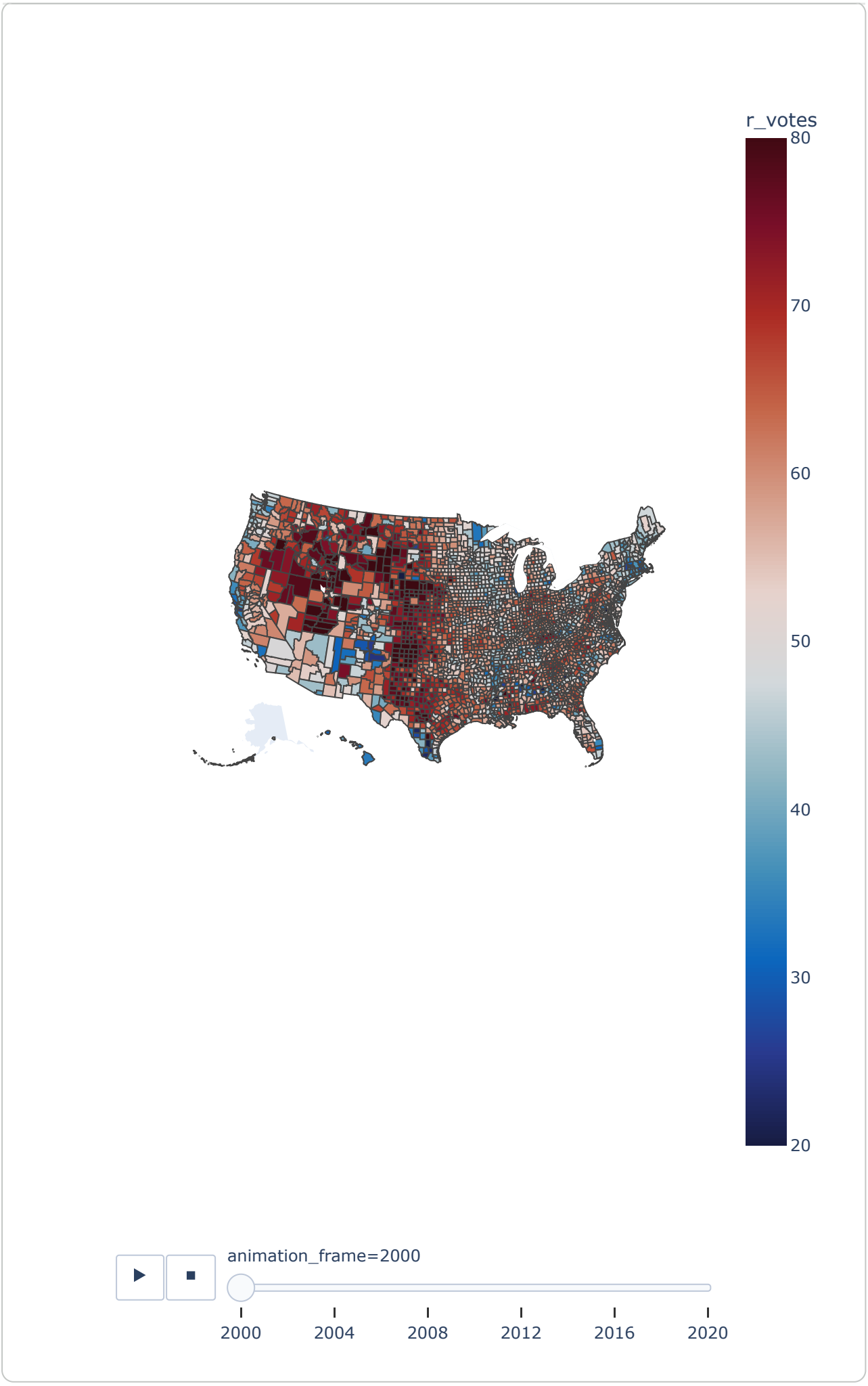
```
plot_sample=counties[counties['year']>2007] # subset the data to only include

px.choropleth( # plot a choropleth map using the plotly express (px) library
               plot_sample, # load the dataframe
               locations='county_fips', # set the location column to the sta
               geojson=county_polygons, # set the location mode to USA state
               scope='usa', # set the scope to the USA, so that it only plot
               color="unemployment", # set the color of the states to corres
               animation_frame=plot_sample["year"].astype(str), # set the an
               color_continuous_scale=px.colors.sequential.Viridis, # set th
               range_color=[0, 10], # set the range of the color scale to 0-
               height=1000) # set the height of the map to 1000 pixels
```


unemployment
10
8
6
4
2
0
animation_frame=2008
2008
2010
2012
2014
2016
2018
2020

This map is interactive-- meaning you can zoom in, pan around, and hover over it to get further information on the unemployment level in each county. You can also use the slider at the bottom to toggle between different years; if you move the slider from 2008 to 2009, you'll see lots of yellow suddenly appearing. A similar thing happens between 2019 and 2020. What's going on? Play around with this map for a second, and make note of spatial and temporal trends in unemployment.

Now we're going to do the same thing for the elections data, which I've taken the liberty of cleaning. Let's load it up as a dataframe called `elections`, and make another map in which we plot vote shares in various elections such that red shows republican support, and blue shows democratic support.

```
elections=pd.read_csv('https://storage.googleapis.com/qm2/wk10/elections.csv'
px.choropleth(
                elections,
                locations='county_fips',
                geojson=county_polygons,
                scope='usa',
                color="r_votes",
                animation_frame=elections["year"].astype(str),
                color_continuous_scale=px.colors.diverging.balance,
                range_color=[20, 80],
                height=1000)
```


r_votes
80
70
60
50
40
30
20
animation_frame=2000
2000
2004
2008
2012
2016
2020

Explore the map above. What do you notice about republican vote share, particularly as it relates to the previous map of unemployment?

Now we've got two datasets-- one on unemployment and another on election results. We want to merge them but CAREFUL: each row corresponds to the value of a variable x in county i and time t (so, x_{it}); for example, the value in the first row of our dataset under the unemployment column would be $unemployment_{01001,2000}$; i.e., the unemployment rate in Atauga County, Alabama (FIPS code 01001), in the year 2000. When our data has this structure (x_{it}), we call it **panel data**. It must be handled differently from **cross sectional data** (x_i), from merging to estimation.

We can't just merge on i or t, we need to merge on both. We can do so as follows:

```
1 df_c=pd.merge(elections,counties, on=['county_fips','year'])
2 df_c.head()
```

	county_fips	year	d_votes	r_votes	state	county	unemployment	p
0	01001	2008	25.773021	73.613637	1	1	5.3	
1	01001	2012	26.587832	72.618252	1	1	7.1	
2	01001	2016	23.769671	72.766588	1	1	5.1	
3	01001	2020	27.018365	71.436802	1	1	5.3	
4	01003	2008	23.811922	75.259479	1	3	4.8	

Next steps: Generate code with df_c | New interactive sheet

Exercise

OK. Our data is clean and ready for analysis. Because we're going to be investigating the relationship between unempoyment rates and republican voteshare via a regression model, we're going to need to follow the four steps of regression modeling from last week.

First, formulate a research question (complete with null and alternative hypothesis), and then follow these steps for our dataset, `df_c` (bonus points if you account for the influence of population).

1. Summary Statistics
    - Table of Summary Statistics
2. Visualisation
    - Exploratory Plots

3. Assumptions
    - A. Independence
    - B. Heteroscedasticity: Regression plots + Q-Q plot
    - C. Multicollinearity: VIF + Correlation Matrix
4. Regression
    - Regression Table

For the moment, when you run the regression, ignore the fact that we have panel data and just run a regular regression of the form

$$Y = \beta_0 + \beta_1 X + \epsilon$$

Accounting for Space and Time

If you've done things correctly, you'll notice two things. First, the appears to be a generally negative relationship between unemployment and republican voteshare; in other words, places with higher unemployment tend to vote *against* republicans. Second, we've egregiously violated the independence assumption. We have repeat observations of the same individuals (counties) over time. As such, this result may be biased unless we account for space and time.

As we saw in the lecture, panel data actually contains *two* sources of variation: differences *between* individuals (in this case, counties), and *within* individuials. So, a simple research question such as "Does unemployment increase republican voteshare" is actually two different questions:

1. Does a higher level of unemployment lead to higher republican vote shares **between counties**?
2. Does an *increase* in the unemployment rate over time lead to an *increase* in republican vote shares **within counties**?

Neither is more important than the other, but we must be careful not to conflate them as they are very different questions. A straighforward way of answering the first question would be to get rid of the time dimension in our data by running a separate regression for each year:

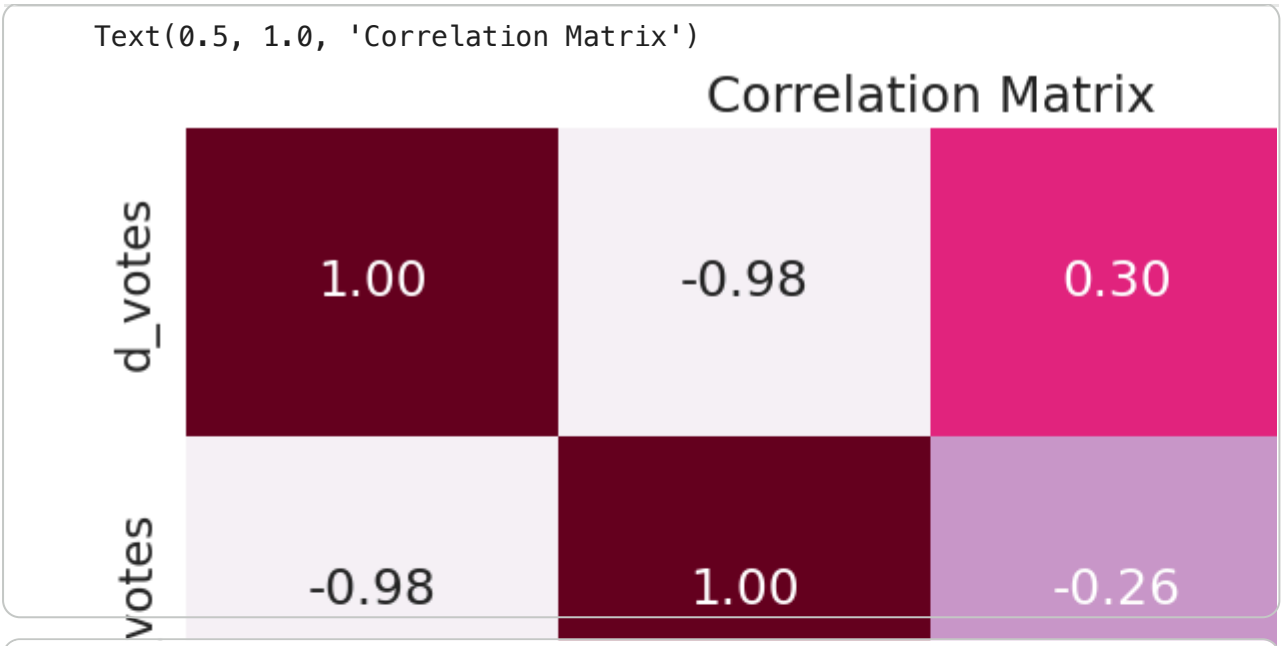
```
model = ols('r_votes ~ unemployment + population', data=df_c).fit()

models = [model]
names = ['Unemployment + Population']

table = summary_col(
    results=models,
    stars=True,
    float_format='%0.3f',
    model_names=names,
    info_dict={
        'N':       lambda x: f'{int(x.nobs)}',
```

```
        'R2':  lambda x: f'{x.rsquared:.3f}',
        'R2_adj':  lambda x: f'{x.rsquared_adj:.3f}'
    }
)

print(table)
```

```
==========================================
               Unemployment + Population
------------------------------------------
Intercept      72.298***
               (0.355)
unemployment   -1.535***
               (0.052)
population     -0.028***
               (0.001)
R-squared      0.154
R-squared Adj. 0.153
N              12457
R2             0.154
R2_adj         0.153
==========================================
Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01
```

```
#Multicollinearity: Correlation Matrix

sns.heatmap(df_c[['d_votes', 'r_votes', 'unemployment', 'population']].corr()
            annot=True,
            fmt=".2f",
            cmap='PuRd')

plt.title('Correlation Matrix')
```

```
Text(0.5, 1.0, 'Correlation Matrix')
```



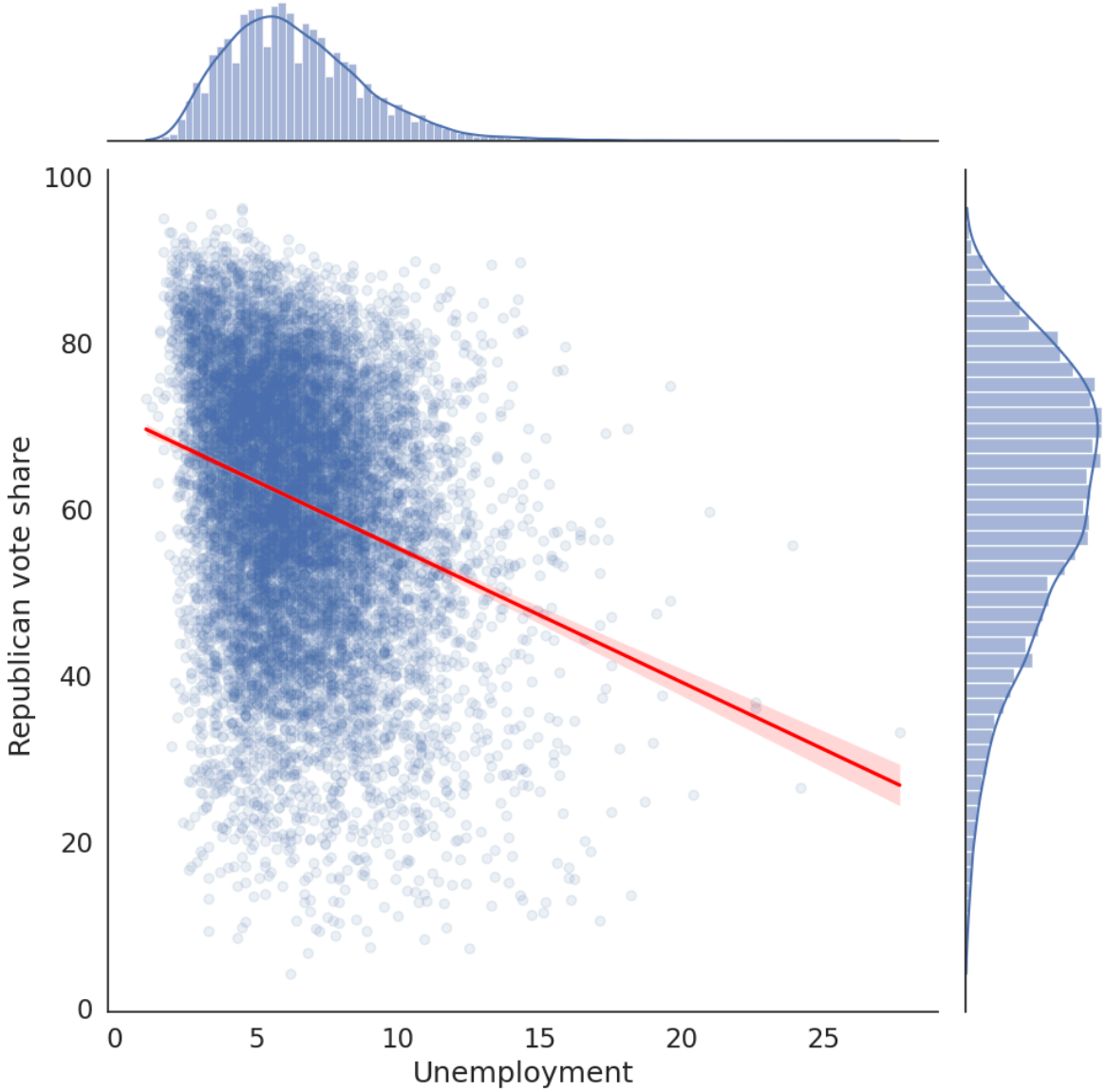

```
# Multicollinearity: calculating VIF
# This function is amended from: https://stackoverflow.com/a/51329496/4667568

from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant

def drop_column_using_vif_(df, list_var_not_to_remove=None, thresh=5):
    '''
    Calculates VIF each feature in a pandas dataframe, and repeatedly drop th
    A constant must be added to variance_inflation_factor or the results will

    :param df: the pandas dataframe containing only the predictor features, n
    :param list_var_not_to_remove: the list of variables that should not be r
    :param thresh: the max VIF value before the feature is removed from the d
    :return: dataframe with multicollinear features removed
    '''
    while True:
        df_with_const = add_constant(df)

        vif_df = pd.Series([variance_inflation_factor(df_with_const.values, i
               for i in range(df_with_const.shape[1])], name= "VIF",
              index=df_with_const.columns).to_frame()


        vif_df = vif_df.drop('const')


        if list_var_not_to_remove is not None:
            vif_df = vif_df.drop(list_var_not_to_remove)

        print('Max VIF:', vif_df.VIF.max())

        if vif_df.VIF.max() > thresh:
            index_to_drop = vif_df.index[vif_df.VIF == vif_df.VIF.max()].toli
            print('Dropping: {}'.format(index_to_drop))
            df = df.drop(columns = index_to_drop)
        else:

            break

    return df

ind_vars = ['d_votes', 'unemployment', 'population']
```

```
ind_vars=['r_votes', 'unemployment', 'population']

vif = drop_column_using_vif_(df=df_c[ind_vars], thresh=5)
print("The columns remaining after VIF selection are:")
print(vif.columns)
```

```
Max VIF: 1.181431829126759
The columns remaining after VIF selection are:
Index(['r_votes', 'unemployment', 'population'], dtype='object')
```

```
#Homoscedasticity

sns.jointplot(df_c,
              x='unemployment',
              y='r_votes',
              kind="reg",
              scatter_kws=dict(alpha=0.1),
              line_kws=dict(color='red'),
              height=10)

plt.xlabel('Unemployment')
plt.ylabel('Republican vote share ')
```

```
Text(69.75, 0.5, 'Republican vote share ')
```



```
from statsmodels.formula.api import ols
from statsmodels.iolib.summary2 import summary_col

model= ols('r_votes ~ unemployment + population', data=df_c).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 r_votes   R-squared:
Model:                             OLS   Adj. R-squared:
Method:                  Least Squares   F-statistic:
Date:                 Tue, 13 Jan 2026   Prob (F-statistic):
Time:                         15:20:33   Log-Likelihood:
No. Observations:               12457   AIC:                             1
Df Residuals:                   12454   BIC:                             1
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025
------------------------------------------------------------------------------
Intercept      72.2981      0.355    203.732      0.000      71.603
unemployment   -1.5355      0.052    -29.688      0.000      -1.637
population     -0.0282      0.001    -35.920      0.000      -0.030
==============================================================================
Omnibus:                      475.411   Durbin-Watson:
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
Skew:                          -0.343   Prob(JB):                        1
Kurtosis:                       4.005   Cond. No.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
```

```
models=[] # create empty list to store the models
names=[] # create empty list to store the names of the models
years=df_c['year'].unique()

for year in years: # loop through years from 2000 to 2020 in increments of 4
    election=df_c[df_c['year']==year] # subset the data to only include the y
    model= ols('r_votes ~ unemployment + population', data=election).fit() #
    models.append(model) # append the model to the list of models
    names.append(str(year)) # append the name of the model to the list of nam

table=summary_col( # create a regression table
    models, # pass the models to the summary_col function
    stars=True, # add stars denoting the p-values of the coefficient to the t
    float_format='%0.3f', # set the decimal places to 3
    model_names=names, # set the names of the model
    info_dict = {"N":lambda x: "{0:d}".format(int(x.nobs))}) # add the number

print(table) # print the table
```

```

=====================================================
                2008      2012      2016      2020
-----------------------------------------------------
```

```
Intercept      68.971*** 73.383*** 70.706*** 81.771***
               (0.680)   (0.727)   (0.788)   (0.803)
unemployment   -1.891*** -1.618*** -1.101*** -2.323***
               (0.109)   (0.088)   (0.142)   (0.115)
population     -0.023*** -0.024*** -0.034*** -0.029***
               (0.001)   (0.002)   (0.002)   (0.002)
R-squared      0.148     0.165     0.142     0.229
R-squared Adj. 0.148     0.165     0.141     0.229
N              3114      3114      3115      3114
=================================================
Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01
```

This table is pretty informative. Using what we learned from last week, we can say that for the 2020 election,

- A 1% increase in the unemployment rate was associated with a 2.3% *decrease* in republican voteshare.
- A 1000-person increase in population was associated with 0.029% decrease in republican voteshare.
- both of these results are statistically signifiant at the 0.01 level.
- 23% of the variation in republican voteshare can be explained by unemployment and population.

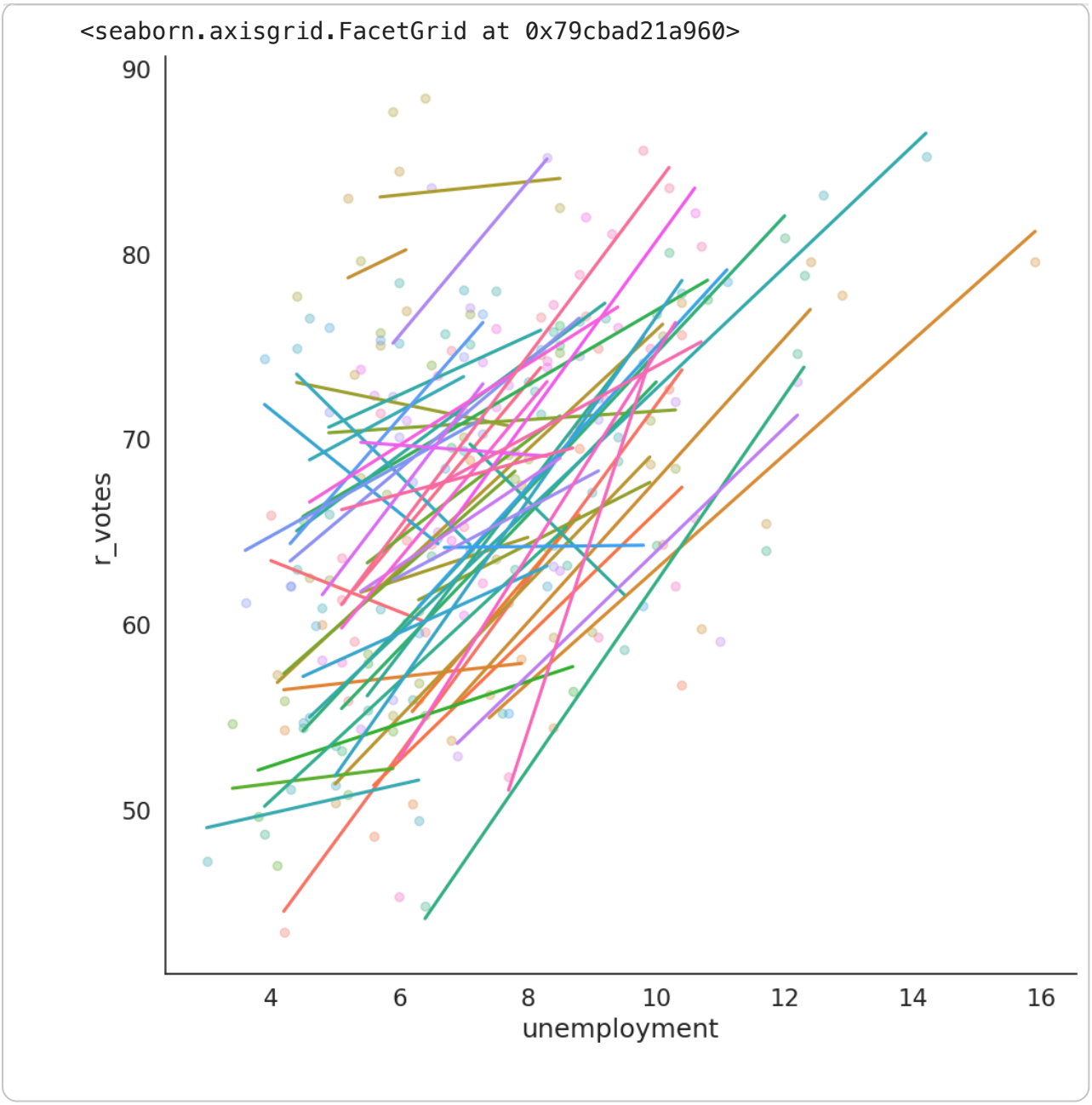
Crucially, "increase" in this context pertains to *differences in between counties*!

We can also compare these results across different elections. The coefficient for the unemployment variable in 2020 is over twice the size of the same coefficient in 2016! So it looks like actually unemployment and republican voteshare are *negatively* related, contrary to popular belief.

But is this the whole story?

Below, i've isolated West Virginia, one of the states with the highest unemployment rates in America. Instead of drawing a new regression line every year, i've drawn a new regression line for each county over the six elections.

```
1 west_virginia=df_c[(df_c['state']==54)]
2 sns.lmplot(data=west_virginia, x='unemployment', y='r_votes', ci=None, hue='c
```



Within a given county, an increase in the unemployment rate is associated with an **increase** in republican voteshare! This is where the second question comes in (variation within counties).

We got away with doing a series of cross-sectional analyses (a new regression for each election) because we have over 3000 counties, so $n > 3000$ for each of those regressions (though even so, we're still splitting our data up and it would be better to leverage the full dataset of >18000 observations in one regression). It also provides relatively useful information about the importance of unemployment across the country for each election. We can't really apply the same thinking to this situation, since we only have six time periods. If we ran a separate regression for each county, we would only have six observations per regression-- nowhere near enough to satisfy the central limit theorem (at least n>30). The insights would also be of limited utility; we would get over 3000 unique estimates for the realtionship between county-level employment and election results. Imagine trying to fit *that* into one table.

Luckily, there's a way of modeling this relationship that allows us to account for differences in between counties, while also capturing the variation within counties. This is called a **Fixed Effect regression**

> **Fixed Effects Models**: In experimental research, unmeasured differences between subjects are often controlled for via random assignment to treatment and control groups. Hence, even if a variable like Socio-Economic Status is not explicitly measured, because of random assignment, we can be reasonably confident that the effects of SES are approximately equal for all groups. Of course, random assignment is usually not possible with most survey research. If we want to control for the effect of a variable, we must explicitly measure it. If we don't measure it, we can't control for it. In practice, there will almost certainly be some variables we have failed to measure (or have measured poorly), so our models will likely suffer from some degree of omitted variable bias. When we have panel data (the same people/states/counties. etc. measured at two or more points in time) another alternative presents itself: we can use the subjects as their own controls. With panel data we can control for stable characteristics (i.e. characteristics that do not change across time) whether they are measured or not. These include such things as sex, race, and ethnicity for individuals, or urban/rural, topography, economic structure for geographic areas. The idea is that, whatever effect these variables have at one point in time, they will have the same effect at a different point in time because the values of such variables do not change.

A fixed effect regression takes the following form:

$$Y_{it} = \alpha_i + \beta X_{it} + \epsilon_{it}$$

Where:

- X_{it} are the independent variables (e.g. population and unemployment) whose values vary over time.
- β is the slope coefficient for variable x (e.g. unemployment). The model assumes that these effects are time-invariant, e.g. the effect of x is the same at same 1 as it is at time 4 (although the value of x can be different at different time periods).
- α_i and ϵ_{it} are both error terms. ϵ_{it} is different for each individual at each point in time. α_i only varies across individuals but not across time. We can think of α_i as representing the effects of all the time invariant/stable variables that have NOT been included in the model. So, given that we have 6 time periods for each county then the six records for county 1 would all have the same value for α_1,

the six records for county 2 would all have the same value for α_2, etc. But, ϵ_{it} is free to be different for every case at every time period.

A fixed effect regression allows us to account for α_i through a technique called **demeaning**

> **Demeaning**: After demeaning, all variables for all cases have a mean of 0. That means that all the between-subject variability has been eliminated. All that is left is the within-subject variability. So, with a fixed effects model, we are analyzing what causes individual's values to change across time. Variables whose values do not change (like race or gender) cannot cause changes across time (unless their effects change across time as well). However, whatever effect they have at one time is the same effect that they have at other times, so the effects of such stable characteristics are controlled.

In essence, you can picture this as allowing you to draw a separate regression line through each set of observations from the same group in your data (in this case, one county over time); however, while the *intercept* of these lines can vary (their absolute position), they will all have the same *slope* and will therefore be parallel. This is important, as we want to find one slope-- one common effect of x-- that fits *all* groups.

Run the command below to install the library.

```
1 !pip install linearmodels
```

```
Collecting linearmodels
  Downloading linearmodels-7.0-cp312-cp312-manylinux2014_x86_64.manyli
Requirement already satisfied: numpy<3,>=1.22.3 in /usr/local/lib/pyth
Requirement already satisfied: pandas>=1.4.0 in /usr/local/lib/python3
Requirement already satisfied: scipy>=1.8.0 in /usr/local/lib/python3.
Requirement already satisfied: statsmodels>=0.13.0 in /usr/local/lib/p
Collecting mypy_extensions>=0.4 (from linearmodels)
  Downloading mypy_extensions-1.1.0-py3-none-any.whl.metadata (1.1 kB)
Collecting pyhdfe>=0.1 (from linearmodels)
  Downloading pyhdfe-0.2.0-py3-none-any.whl.metadata (4.0 kB)
Collecting formulaic>=1.2.1 (from linearmodels)
  Downloading formulaic-1.2.1-py3-none-any.whl.metadata (7.0 kB)
Collecting interface-meta>=1.2.0 (from formulaic>=1.2.1->linearmodels)
  Downloading interface_meta-1.3.0-py3-none-any.whl.metadata (6.7 kB)
Requirement already satisfied: narwhals>=1.17 in /usr/local/lib/python
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/
Requirement already satisfied: wrapt>=1.0 in /usr/local/lib/python3.12
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/li
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/pytho
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/d
```

```
Downloading linearmodels-7.0-cp312-cp312-manylinux2014_x86_64.manylinu
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.5/1.5 MB 14.5 MB/s eta 0
Downloading formulaic-1.2.1-py3-none-any.whl (117 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 117.3/117.3 kB 3.6 MB/s et
Downloading mypy_extensions-1.1.0-py3-none-any.whl (5.0 kB)
Downloading pyhdfe-0.2.0-py3-none-any.whl (19 kB)
Downloading interface_meta-1.3.0-py3-none-any.whl (14 kB)
Installing collected packages: mypy_extensions, interface-meta, pyhdfe
Successfully installed formulaic-1.2.1 interface-meta-1.3.0 linearmode
```

```
1 from linearmodels import PanelOLS
2 from linearmodels import RandomEffects
3 import statsmodels.formula.api as smf
4 from linearmodels.panel import compare
5
6 df_c=df_c.set_index(['county_fips','year']) # set the index to the county fip
7 panel = PanelOLS.from_formula('r_votes ~ 1  + population + unemployment  + En
8 print(compare({'Fixed Effects': panel,}, stars=True)) # print the model forma
```

```
            Model Comparison
=====================================
                        Fixed Effects
-------------------------------------
Dep. Variable                 r_votes
Estimator                    PanelOLS
No. Observations                12457
Cov. Est.                   Unadjusted
R-squared                       0.0206
R-Squared (Within)              0.0206
R-Squared (Between)            -0.1061
R-Squared (Overall)            -0.0939
F-statistic                     98.465
P-value (F-stat)                0.0000
====================    ============
Intercept                   66.818***
                             (159.05)
population                 -0.0727***
                            (-10.241)
unemployment               -0.3268***
                            (-9.3595)
====================== ==============
Effects                         Entity
-------------------------------------

T-stats reported in parentheses
```

When accounting for time-invariant differences between counties, the effect of population remains negative. This suggests that counties in which the population is *decreasing* tend to experience an increase in republican voteshare. More specifically, for every 1000 people that leave a county, republican voteshare increases by 0.06%.

The really interesting part of this regression table, however, is the coefficient on the unemployment variable, which is now positive. This suggests that-- once we account for the differences between counties-- an increase in the unemployment rate *within* a

county is *positively* associated with republican voteshare. Indeed, a 1% increase in the unemployment rate leads to a 0.28% increase in republican voteshare.

This regression output even gives us three separate R^2 values-- one for between-variation, another for within, and one overall.

---

2. Difference in Differences

One of the reasons that we observe a signficant relationship between unemployment and voting behaviour in last week's workshop is that the Republican and Democratic parties have opposing views on what to do about unemployment. Democratic lawmakers have historically been in favour of increasing the minimum wage to benefit low-income workers, while Republicans have generally opposed this on the basis that it would hurt these very workers by increase unemployment. Indeed, classical economic theory holds that an increase in wages would lead to a reduction in employment; A business that makes $100k in revenue per year and spends all of it on employing 20 people can't suddenly start paying their workers double their salaries-- unless it fires half of its workers. This is obviously a simplified model though-- minimum wage laws typically don't double wages, and businesses don't operate at-cost, they turn a profit which they could use to pay their workers more. In the rest of this workshop, we're going to be investigating this question empirically:

Do minimum wage laws increase unemployment?

Note that this is a *causal* question; i'm not asking if they're correlated-- i'm asking if one causes the other. The burden of proof here is much higher than observing correlations, and we have to think seriously about **endogeneity**. In partiuclar, we need to account for the influence of omitted variables (e.g. a recession, or the economic composition of a state), the potential for reverse causality (states implementing minimum wage laws in response to unemployment crises), and selection bias.

In a lab, you can conduct causal inference by running an experiment. You can randomly select individuals, split them into a control group and a treatment group, measure their values in an outcome variable prior to a treatment, administer a treatment, and measure their respective values after the treatment. If you observe a change in the outcome variable in the treatment group after having administered the treatment, you can interpert that as the causal effect of treatment. This is because we're able to make a plausible argument that the **control group can act as a counterfactual (a stand-in) for the treatment group in the absence of treatment**. Both groups had the same values before the treatment, then the only

thing that changed between them was the treatment, so if we observe a change in the outcome variable, it must be due to treatment.

In the real world, we rarely get to run expermients of this kind. Instead, we have to hunt for **natural experiments**: situations in which there is a **treatment** which we're interested in measuring the effect of, and two groups that can plausibly act as a treatment and control group.

> **Difference in Difference** is a quasi-experimental design that makes use of longitudinal data from treatment and control groups to obtain an appropriate counterfactual to estimate a causal effect. DID is typically used to estimate the effect of a specific intervention or treatment (such as a passage of law, enactment of policy, or large-scale program implementation) by comparing the changes in outcomes over time between a population that is enrolled in a program (the intervention group) and a population that is not (the control group).

The Difference in Difference model can be estimated as a simple regression model of the following form:

$$Y_{it} = \beta_0 + \beta_1 Treatment_i + \beta_2 Post + \beta_3 (Treatment_i \times Post_t) + \varepsilon_{it}$$

- $Treatment_i$ is 0 for the control group and 1 for the treatment group
- $Post_t$ is 0 for before and 1 for after

we can insert the values of $Treatment$ and $Post$ using the table below and see that coefficient (β_3) of the interaction of $Treatment$ and $Post$ is the Difference in Differences (DID) estimator:

Card and Krueger (1994) found one such natural experiment, allowing them to estimate the causal effect of an increase in the state minimum wage on unemployment using a DiD model; In 1992, New Jersey raised the state minimum wage from $4.25 to 5.05 while the minimum wage in neighbouring Pennsylvania stayed the same at $4.25.

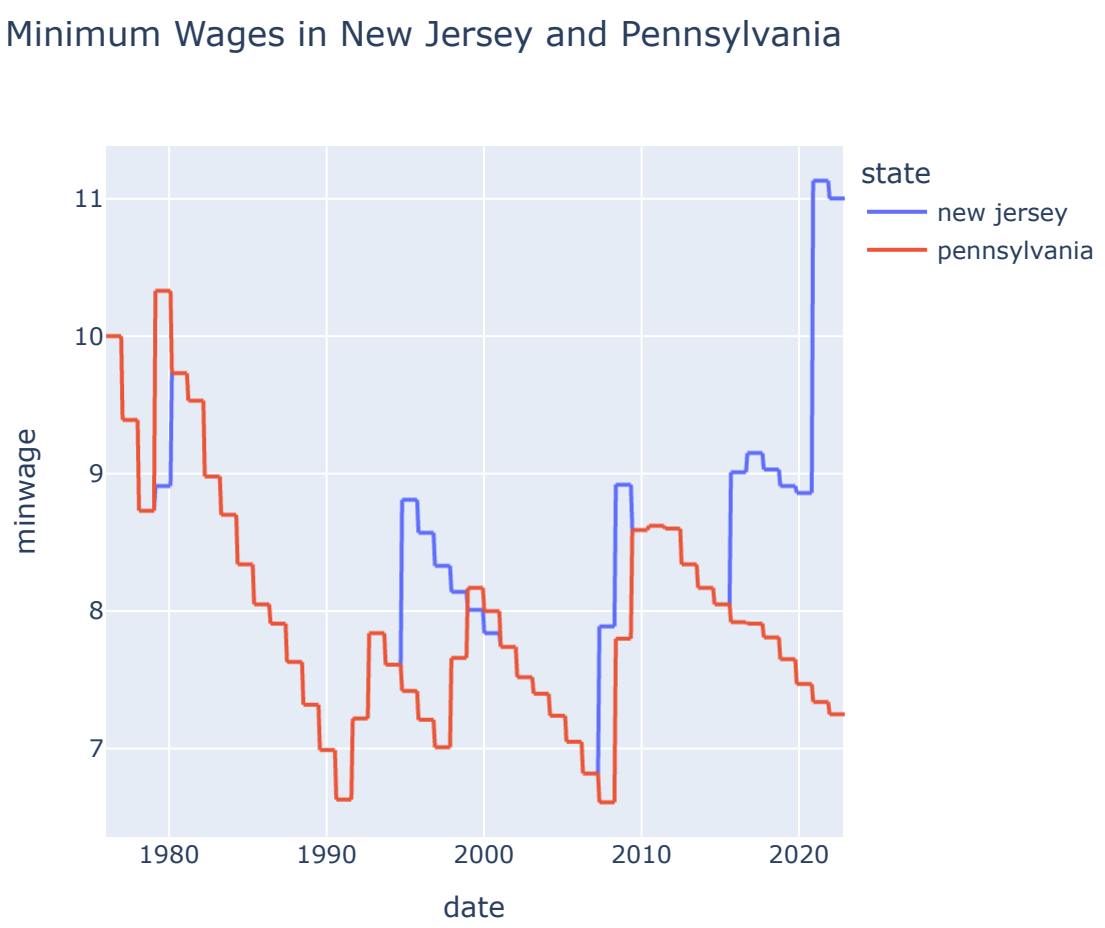
- Treatmeng Group: New Jersey
- Control Group: Pennsylvania
- Pre-Treatment Period: before 1992
- Post-Treatment Period: after 1992

They conducted a survey of 384 fast-food restaurants across both states, right before and right after the law came into effect in New Jersey, asking them how many people they employed. They ran a Difference-in-Differences model, and found that the coefficient β_3 was positive but not statistically significant. In other words, the average

total employees per restaurant *increased* after the minimum wage increased, but this could have been due to random chance.

That was a long time ago. Things have changed since then, including the fact that we have access to a lot more data and computational power. Let's see if we can replicate Card and Krueger's results with more recent data. I've downloaded data on unemployment, minimum wage levels, and Gross Domestic Product at the state level going back to 1976. Let's have a look at minimum wages in New Jersey and Pennsylvania over time:

```
df_s=pd.read_csv('https://storage.googleapis.com/qm2/wk10/state_data.csv', pa
did=df_s[df_s['state'].isin(['pennsylvania', 'new jersey'])] # subset the dat

px.line(did, x='date', y='minwage', color='state', title="Minimum Wages in Ne
```



The plot above sort of looks like a set of descending staircases; this is for two reasons. The plateaus exist because each row in the dataframe `df_s` is the value of a state in a given *month*, but we only have minimum wage data for every *year*. So we get 12 consecutive values of minimum wage every year. The reason that the staircases are descending is because these minimum wages are adjusted for inflation. No matter where you're from, you've probably heard a grandparent say something

along the lines of "My parents would send me to the shops with 25 cents to buy groceries for the week", but now it costs £9 for a bag of chips. That's inflation-- every year things tend to get slightly more expensive, so if the same *absolute* minimum wage actually diminishes in "real" terms, which is what the variable `minwage` measures. Incidentally, this is one of the main reasons University staff have been on strike. Anyway. Back to minimum wages.

This plot shows that for the past fifty years, New Jersey and Pennsylvania have had largely similar minimum wage policies. There have been a couple moments of divergence, including in the 1990s when the Card and Krueger study was conducted. However, the biggest divergence actually started taking place in 2014 when New Jersey seems to have begun taking a wildly different approach. While Pennsylvania has had the same minimum wage since 2008 (and therefore seen a decline in inflation-adjusted wages), New Jersey has raised the minimum wage significantly twice. In 2020, New Jersey's minimum wage was around 50% higher than Pennsylvania's. We can exploit the fact that these two states have historically had similar minimum wage laws but have recently experienced a big divergence to see if that change in minimum wages has resulted in a change in employment levels.

Our Difference-in-Differences setup is as follows:

$$Unemployment_{state,year} = \beta_0 + \beta_1 Treatment_{state} + \beta_2 Post_y + \beta_4 GDP_{state,year} + \varepsilon_{it}$$

- New Jersey is the **treatment group**
- Pennsylvania is the **control group**
- Years before 2014 is the **pre-treatment period**
- Years after 2014 is the **post-treatment period**

```
did['post']=np.where(did['date']>='2014-01-01',1,0) # create a variable that
did['treatment']=np.where(did['state']=='new jersey',1,0) # create a variable
did['post_treatment']=did['post']*did['treatment'] # create a variable that i
```

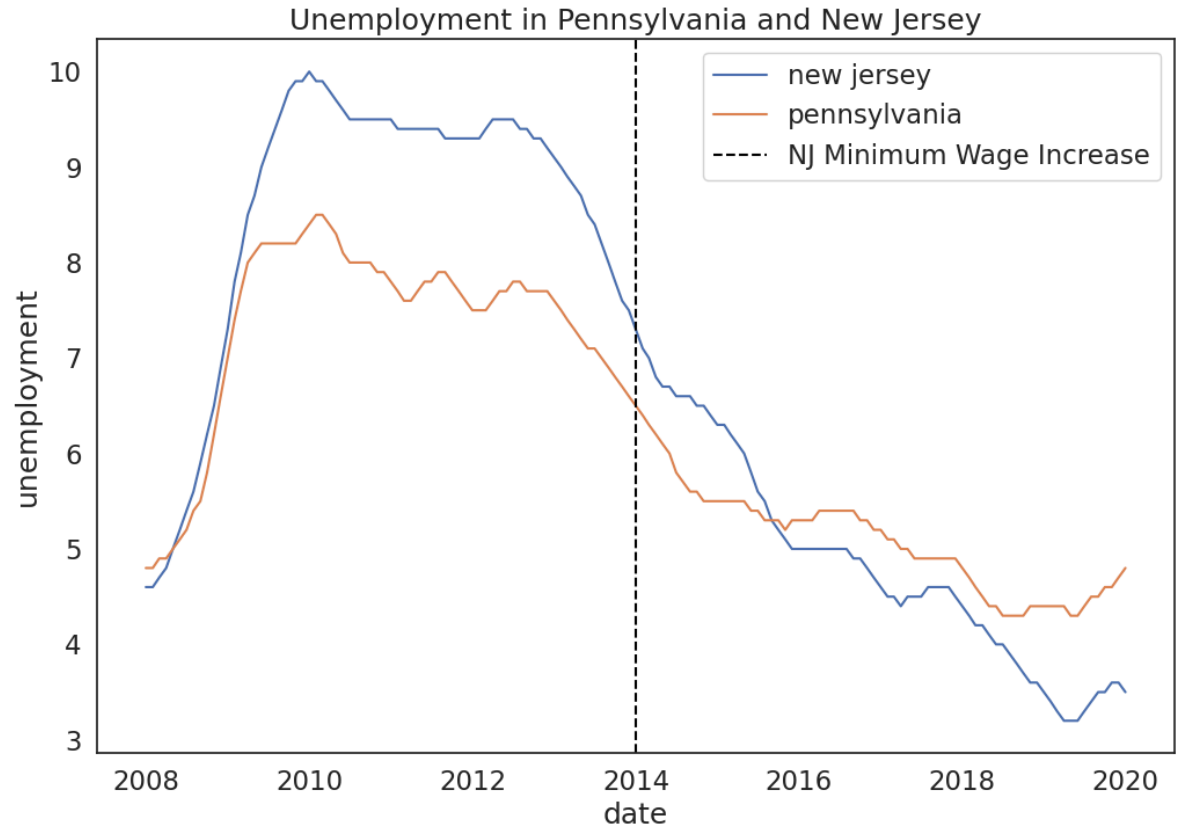
Before we proceed with the analysis, though, we need to satisfy two assumptions that will allow us to argue that Pennsylvania can act as a valid control group for New Jersey:

1. No simultaneous treatments:
    - If, for example, New Jersey suddenly entered a massive recession in 2014 as well, we couldn't really argue that resulting effects on employment are due solely to the minimum wage law. To account for this, we'll be including state-level GDP as an additional independent variable in our DiD model.
2. Parallel Trends:

- Both states have to have been experiencing similar trends in the **dependent variable** (unemployment) prior to the treatment (minimum wage law). If they were trending in opposite directions for unobserved reasons, ensuing differences in unemployment may be due to those unobserved reasons rather than the treatment.
- We can check this by plotting the dependent variable for both groups over time, and indicating the timing of the treatment.

```
did=did[(did['date']>='2008-01-01') & (did['date']<='2020-01-01')]
sns.lineplot(data=did,x='date',y='unemployment',hue='state')
plt.axvline(pd.to_datetime('2014-01-01'),color='black',linestyle='dashed', la
plt.title('Unemployment in Pennsylvania and New Jersey')
plt.legend()
```

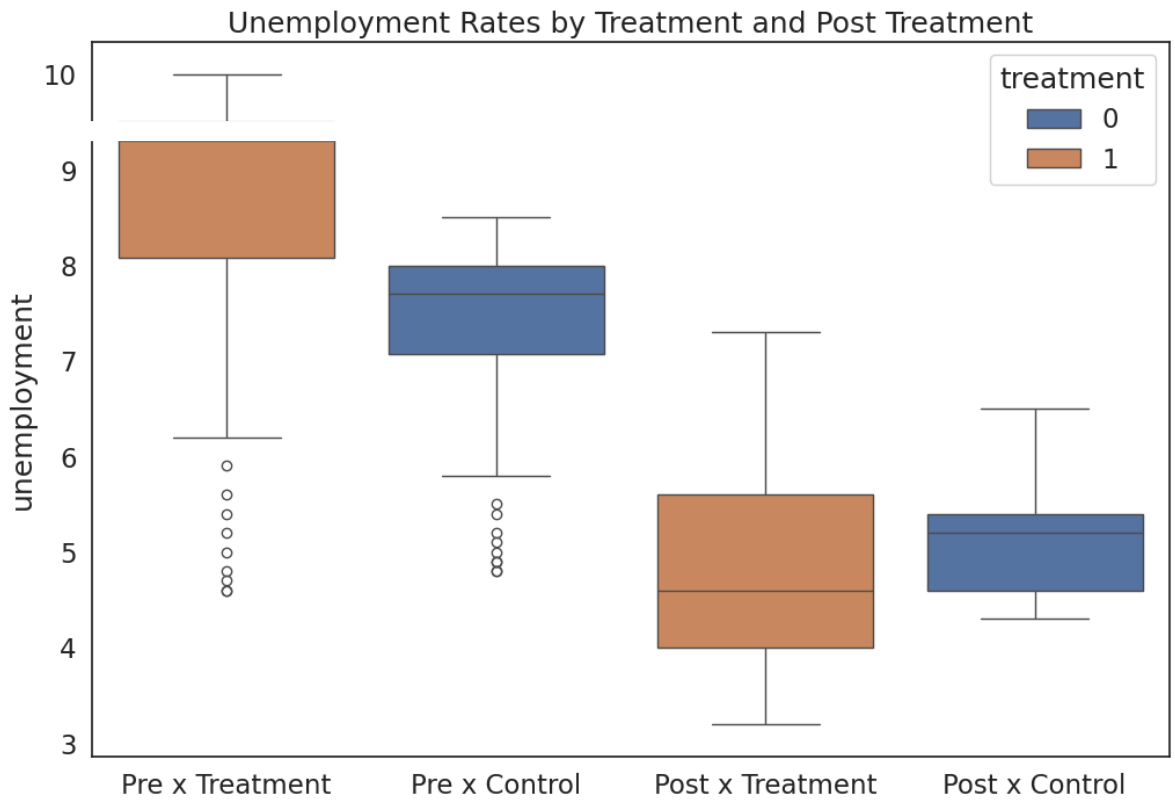
```
<matplotlib.legend.Legend at 0x79cba18e88f0>
```



This plot shows a big spike in unemployment occurring for both Pennsylvania and New Jersey as a result of the 2008 financial crisis. New jersey had a higher unemployment rate than Pennsylvania, but their trends are largely parallel and decreasing after 2012. In the years following the minimum wage law, New Jersey's unemployment rate actually dips below Pennsylvania's for the first time in years. Let's look at this in the form of boxplots:

```
did['category']=did['treatment'].astype(str)+did['post'].astype(str) # this v
sns.boxplot(x='category', y='unemployment', hue='treatment', data=did).set_xt
plt.xlabel('')
```

```
plt.title('Unemployment Rates by Treatment and Post Treatment')
plt.show()
```



This plot is fascinating in and of itself. The two box plots on the left show the unemployment values of the counties prior to the minimum wage law in 2014, while the two on the right show their values after the minimum wage increases. Pennsylvania (the "control" group) is colored in blue, and New Jersey (the "treatment" group) is colored orange. Prior to the minimum wage increase in 2014, Pennsylvania (blue) has a lower unemployment rate than New Jersey (orange). In the years following New Jersey's passage of the minimum wage law, New Jersey actually has a *lower* unemployment rate than Pennsylvania! This is the only boxplot where the "treatment" (a minimum wage law) is being applied, and it has the lowest unemployment rate.

Let's see if this difference is statistically signfiicant, and calculate a treatment effect:

```
did_model = ols('unemployment ~  post + treatment + post_treatment', did).fit
print(did_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           unemployment   R-squared:
Model:                            OLS   Adj. R-squared:
Method:                 Least Squares   F-statistic:
Date:                Tue, 13 Jan 2026   Prob (F-statistic):
Time:                        15:23:15   Log-Likelihood:
No. Observations:                 290   AIC:
Df Residuals:                     286   BIC:
Df Model:                           3
```

```
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025
------------------------------------------------------------------------------
Intercept      7.3319      0.131     55.810      0.000       7.073
post          -2.2443      0.185    -12.121      0.000      -2.609
treatment      1.1972      0.186      6.444      0.000       0.832
post_treatment -1.4232      0.262     -5.435      0.000      -1.939
==============================================================================
Omnibus:                       54.329   Durbin-Watson:
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
Skew:                          -1.068   Prob(JB):
Kurtosis:                       4.675   Cond. No.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
```

There are some really interesting results from this model-- let's interpret the coefficients one by one.

- `gdp`: GDP is inversely related to unemployment. This makes sense: GDP basically measures the total amount of economic activity, so more economic activity = more employment.
- `post`: this coefficient is negative, but statistically insignificant at the 0.05 level; it indicates that unemployment *generally* decreased for both groups, but that this could be due to random chance.
- `treatment`: again negative but insignficant, meaning that there is no significant difference in unemployment levels between NJ and PA over the entire period.
- `post_treatment`: this is our difference-in-differences estimator, and reflects the causal effect of treatment. It is negative and statistically significant. If we believe that the asusmptions of our model are satisfied, we can claim that:
    - **The introduction of a minimum wage in New Jersey led to a 1.95% decrease in unemployment relative to Pennsylvania**

This is a bold claim. We should do our best to back it up. Notice that i've sort of arbitrarily chosen a window of dates around the minimum wage law-- maybe this result is a fluke, due to the timespan ive chosen.

To address this concern, I'll run the same model 10 times, starting with a really small time window-- just one year on either side of the law-- and progressively expand it.

```
models=[] # create empty list to store the models
names=[] # create empty list to store the names of the models

for window in range(1,10): # loop through years from 2000 to 2020 in incremen
    did=df_s[(df_s['date']>=str(2014-window)+'-01-01') & (df_s['date']<=str(2
    did['post']=np.where(did['date']>='2014-01-01',1,0) # create a dummy vari
```

```
    did['treatment']=np.where(did['state']=='new jersey',1,0) # create a dumm
    did['post_treatment']=did['post']*did['treatment'] # create an interactio
    did_model = ols('unemployment ~ gdp+ post + treatment + post_treatment',

    models.append(did_model) # append the model to the list of models
    names.append('± '+str(window)+' Year') # append the name of the model to

table=summary_col( # create a regression table
    models, # pass the models to the summary_col function
    stars=True, # add stars denoting the p-values of the coefficient to the t
    float_format='%0.3f', # set the decimal places to 3
    model_names=names, # set the names of the model
    info_dict = {"N":lambda x: "{0:d}".format(int(x.nobs))}) # add the number

print(table) # print the table

```

```
=============================================================================
               ± 1 Year  ± 2 Year  ± 3 Year  ± 4 Year  ± 5 Year  ± 6 Y
-----------------------------------------------------------------------------
Intercept      21.423** 33.130*** 24.228*** 22.751*** 19.711*** 15.651
               (9.399)  (2.324)   (1.333)   (0.876)   (0.746)   (1.277
gdp            -0.000   -0.000*** -0.000*** -0.000*** -0.000*** -0.000
               (0.000)  (0.000)   (0.000)   (0.000)   (0.000)   (0.000
post           -0.619   0.100     -0.262    -0.256*   -0.483*** -0.585
               (0.411)  (0.194)   (0.162)   (0.140)   (0.154)   (0.307
treatment      -1.604   -3.686*** -1.850*** -1.466*** -0.903*** -0.364
               (1.896)  (0.479)   (0.285)   (0.190)   (0.169)   (0.295
post_treatment -0.837** -1.707*** -1.848*** -2.108*** -2.111*** -1.953
               (0.323)  (0.158)   (0.134)   (0.115)   (0.130)   (0.258
R-squared      0.856    0.934     0.938     0.952     0.933     0.704
R-squared Adj. 0.844    0.931     0.936     0.951     0.932     0.699
N              50       98        146       194       242       290
=============================================================================
Standard errors in parentheses.
* p<.1, ** p<.05, ***p<.01
```

The row we're mainly interested in is the `post_treatment` coefficient, the treatment effect. It remains significant and negative in all time periods smaller than 8 years, after which point it becomes insignificant;

How do you think this affects our conclusion?

Assessed Question

1. Find the state with the largest increase in minimum wage in the 1990s. This will be our treatment group.
2. Set a 5 year window on either side of the treatment date
3. Make a parallel trends plot using Arizona as a control group
4. Control for the cost of living using the CPI variable.
5. Run a difference in differences model.

What was the percentage change in unemployment that resulted from the introduction of a minimum wage in this case? Is the difference statistically significant?

```
import numpy as np
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.formula.api import ols
from datetime import datetime

# Set style for better visualizations
plt.style.use('seaborn-v0_8-darkgrid')
sns.set_palette("colorblind")

# 1. Initial Data Inspection
print("=== Initial Data Inspection ===")
print(f"DataFrame shape: {df_s.shape}")
print("\nFirst 5 rows:")
print(df_s.head())

print(f"\nCPI Statistics:")
print(f"Maximum CPI: {df_s['cpi'].max():.2f}")
print(f"Minimum CPI: {df_s['cpi'].min():.2f}")
print(f"CPI Range: {df_s['cpi'].max() - df_s['cpi'].min():.2f}")

# 2. Prepare DID Data
print("\n=== Preparing DID Analysis ===")

# Convert date column to datetime if not already
if not pd.api.types.is_datetime64_any_dtype(df_s['date']):
    df_s['date'] = pd.to_datetime(df_s['date'])

# Select treatment and control states
treatment_state = 'iowa'
control_state = 'arizona'

did2 = df_s[df_s['state'].isin([treatment_state, control_state])].copy()
print(f"Selected states: {treatment_state} (treatment) and {control_state} (
print(f"DID dataset shape: {did2.shape}")

# 3. Visualize Minimum Wage
print("\n=== Minimum Wage Visualization ===")
fig_minwage = px.line(
    did2,
    x='date',
    y='minwage',
    color='state',
    title=f"Minimum Wage in {treatment_state.title()} and {control_state.tit
    labels={'date': 'Date', 'minwage': 'Minimum Wage ($)', 'state': 'State'}
    template='plotly_white'
)
fig_minwage.update_layout(
    title_x=0.5,
    hovermode='x unified'
)
fig_minwage.show()

# 4. Create DID Variables
treatment_date = pd.to_datetime('1991-01-01')
```

```
analysis_start = pd.to_datetime('1986-01-01')
analysis_end = pd.to_datetime('1996-01-01')

did2['post'] = (did2['date'] >= treatment_date).astype(int)
did2['treat'] = (did2['state'] == treatment_state).astype(int)
did2['post_treat'] = did2['post'] * did2['treat']

# Filter to analysis period
original_size = len(did2)
did2 = did2[(did2['date'] >= analysis_start) & (did2['date'] <= analysis_end
print(f"\nFiltered data from {analysis_start.date()} to {analysis_end.date()
print(f"Rows removed: {original_size - len(did2)}")
print(f"Final DID dataset shape: {did2.shape}")

# 5. Check balance in DID data
print("\n=== Data Balance Check ===")
balance_check = did2.groupby(['state', 'post']).agg({
    'unemployment': ['mean', 'count'],
    'cpi': 'mean'
}).round(3)
print(balance_check)

# 6. Visualize Unemployment with DID
print("\n=== Unemployment Visualization ===")
fig, ax = plt.subplots(figsize=(12, 6))

sns.lineplot(
    data=did2,
    x='date',
    y='unemployment',
    hue='state',
    style='state',
    markers=True,
    dashes=False,
    ax=ax,
    linewidth=2.5
)

# Add treatment line
ax.axvline(
    x=treatment_date,
    color='red',
    linestyle='--',
    linewidth=2,
    alpha=0.7,
    label=f'{treatment_state.title()} Minimum Wage Increase'
)

# Add shaded treatment period
ax.axvspan(
    treatment_date,
    analysis_end,
    alpha=0.1,
    color='red',
    label='Post-treatment Period'
)

# Customize plot
ax.set_title(
    f'Unemployment in {treatment_state.title()} (Treatment) vs {control_stat
    fontsize=14,
```

```
    fontweight='bold'
)
ax.set_xlabel('Date', fontsize=12)
ax.set_ylabel('Unemployment Rate (%)', fontsize=12)
ax.legend(loc='best', fontsize=10)

# Format x-axis dates
ax.xaxis.set_major_locator(plt.MaxNLocator(10))
fig.autofmt_xdate(rotation=45)

plt.tight_layout()
plt.show()

# 7. Parallel Trends Check (Pre-treatment)
print("\n=== Parallel Trends Assumption Check ===")
pre_treatment = did2[did2['date'] < treatment_date].copy()
pre_trends = pre_treatment.groupby(['state', pd.Grouper(key='date', freq='Y'

print("Annual average unemployment (pre-treatment):")
print(pre_trends.round(3))

# Visual check of parallel trends
fig_trends, ax_trends = plt.subplots(figsize=(10, 5))
pre_treatment.groupby(['state', pd.Grouper(key='date', freq='Q')])['unemploy
    ax=ax_trends,
    marker='o',
    linewidth=2
)
ax_trends.axvline(x=treatment_date, color='red', linestyle='--', alpha=0.5)
ax_trends.set_title('Pre-treatment Unemployment Trends (Quarterly Averages)'
ax_trends.set_ylabel('Unemployment Rate (%)')
plt.tight_layout()
plt.show()

# 8. DID Regression Analysis
print("\n" + "="*50)
print("DIFFERENCE-IN-DIFFERENCES REGRESSION RESULTS")
print("="*50)

# Model 1: Basic DID
did_model_basic = ols('unemployment ~ treat + post + post_treat', data=did2)
print("\n1. Basic DID Model (without controls):")
print(did_model_basic.summary().tables[1])

# Model 2: DID with CPI control
did_model_full = ols('unemployment ~ treat + post + post_treat + cpi', data=
print("\n2. DID Model with CPI control:")
print(did_model_full.summary().tables[1])

# Extract key results
treatment_effect = did_model_full.params.get('post_treat', 0)
treatment_pvalue = did_model_full.pvalues.get('post_treat', 1)

print("\n" + "="*50)
print("KEY INTERPRETATION")
print("="*50)
print(f"Treatment Effect (post_treat coefficient): {treatment_effect:.4f}")
print(f"P-value: {treatment_pvalue:.4f}")

if treatment_pvalue < 0.05:
    significance = "statistically significant"
```

```
else:
    significance = "not statistically significant"

print(f"\nThe minimum wage increase in {treatment_state.title()} is estimate
print(f"{'increased' if treatment_effect > 0 else 'decreased'} unemployment
print(f"and this effect is {significance} at the 5% level.")

# 9. Additional Diagnostic Checks
print("\n" + "="*50)
print("MODDIAGNOSTIC CHECKS")
print("="*50)

# Check for multicollinearity
print("\nCorrelation matrix of independent variables:")
corr_matrix = did2[['treat', 'post', 'post_treat', 'cpi']].corr()
print(corr_matrix.round(3))

# Model diagnostics
print(f"\nR-squared: {did_model_full.rsquared:.3f}")
print(f"Adjusted R-squared: {did_model_full.rsquared_adj:.3f}")
print(f"Number of observations: {did_model_full.nobs}")

# 10. Save results for future reference
results_summary = {
    'treatment_state': treatment_state,
    'control_state': control_state,
    'treatment_date': treatment_date,
    'analysis_period': f"{analysis_start.date()} to {analysis_end.date()}",
    'treatment_effect': treatment_effect,
    'treatment_pvalue': treatment_pvalue,
    'observations': len(did2),
    'rsquared': did_model_full.rsquared
}

print("\n" + "="*50)
print("ANALYSIS COMPLETE")
print("="*50)
```