Question 1

a) Panel data includes information for multiple entities across multiple time periods. It allows observing differences between entities and changes for entities over time periods.

b) Serial correlation explains the relationship between the values of an entity over time periods. It occurs when an entity's values are correlated with its past or lagged values.

c) Time dummy is a binary variable that captures time effects in panel data. It takes a value of 1 for the time period of interest and 0 for all other time periods.

d) Fixed effect is an individual-specific dummy variable designed to capture constant differences across entities.

e) Instrumental variable is used to address endogeneity or the influence of potential confounding variables (omitted in the original regression). To be considered valid, the instrumental variable has to be strongly related to the independent variable, independent with respect to the outcome variable and affect the outcome variable only through its relationship with the independent variable.

f) First-stage regression is a step used in instrumental variable analysis to measure the relationship between the instrument and the independent variable that is affected by endogeneity.

g) Exclusion restriction is an assumption in instrumental variable analysis that holds that the instrument affects the outcome variable only via the channel of the independent variable (is not directly related to the outcome).

h) Differences-in-differences is a causal inference method that measures the causal relationship between the intervention and the outcome by comparing the differences between the treated and untreated entities before and after the intervention. The causal inference in DiD is based on the parallel trends assumption, which holds that both treated and untreated entities would follow parallel trends in the absence of the treatment. In other words, the assumption is that the average change in the outcome for the treated entities would be similar to the change in the outcome for untreated entities had the intervention not occured.

i) Propensity score matching is used to address selection bias when comparing treated and control entities. It helps replicate random assignment of the treatment by matching entities from treatment and control groups based on their propensity scores (chances) of getting treatment. Propensity scores are calculated using observed characteristics of the entities.

Question 2

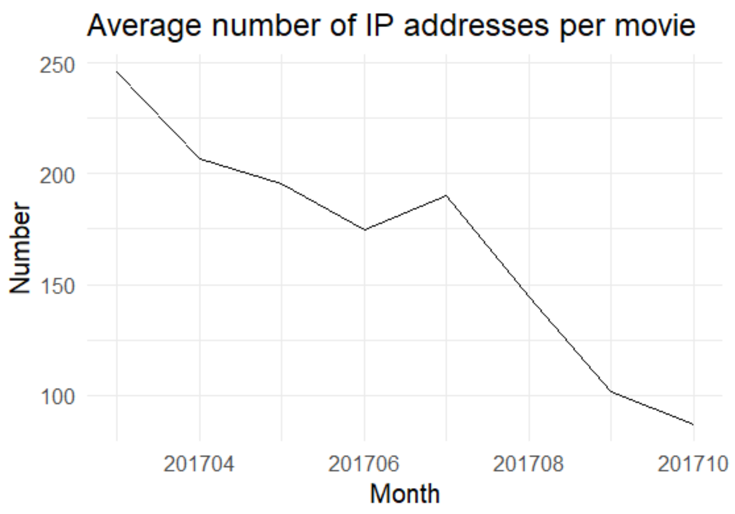
a) This dataset contains 20184 entries and 4 columns. The number of unique movies is 2523 and the included months are March to October 2017.

```
> length(unique(HW.2.a$id))
[1] 2523
> unique(HW.2.a$month)
[1] 201703 201704 201705 201706 201707 201708 201709 201710
```

b) First, we need to convert the logs of the number of IP addresses into actual numbers.

```
> HW.2.a$num_IPs <- exp(HW.2.a$log_IPs)
```

Then we create a dataframe 'avg_mnth_IPs' that represents months and corresponding average number of IP addresses.



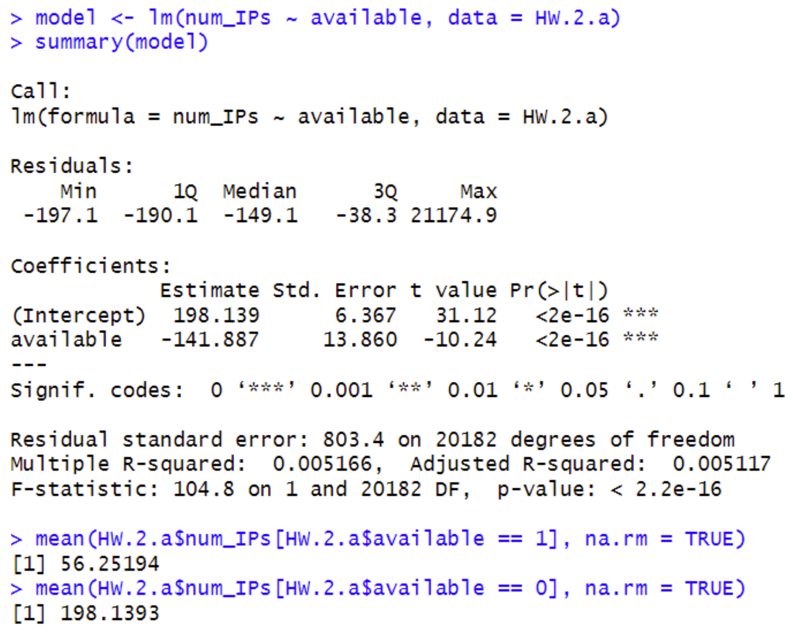
```
> avg_mnth_IPs <- HW.2.a %>% group_by(month) %>% summarise(avg_mnth_IPs = mean(num_IPs, na.rm = TRUE))
> View(avg_mnth_IPs)
```

```
> ggplot(avg_mnth_IPs, aes(x = month, y = avg_mnth_IPs)) +
+     geom_line() + theme_minimal() +
+     labs(title = "Average number of IP addresses per movie",
+          x = "Month", y = "Number")
```

From this plot, we cannot infer any causality between streaming and piracy. Although the downward trend over time is clear, we cannot be sure what causes this reduction looking at this plot. Let's check if there's any statistically significant relationship between availability of a movie on streaming platforms and the number of IP addresses.

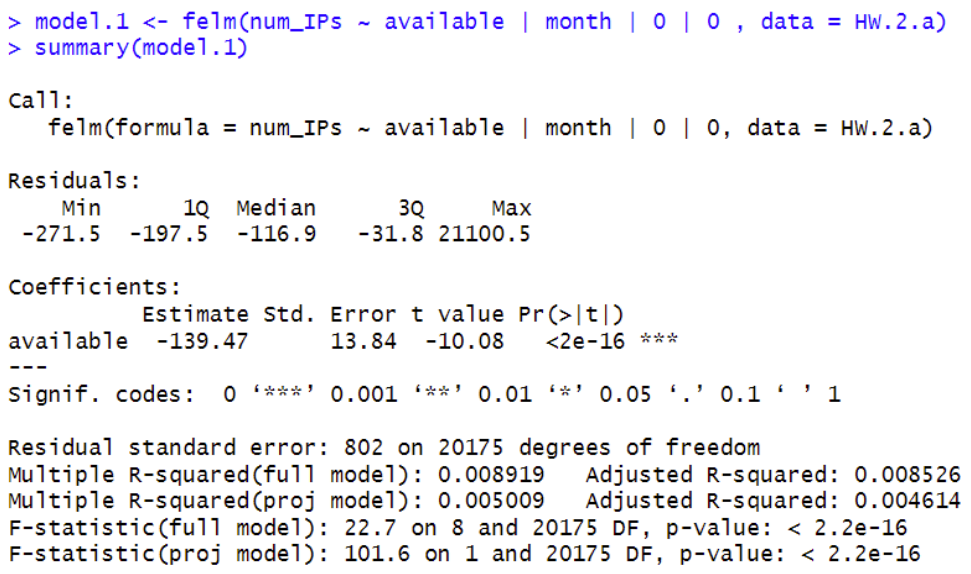
A simple OLS results suggest that on average, all else being equal, availability of a movie on streaming platforms is associated with around 142 less IP addresses exchanging the movie illegally. The coefficient is statistically significantly different from zero.

However, we still cannot be confident about any causality between the two variables. Some omitted variables and/or time and movie-specific differences might also be in play.

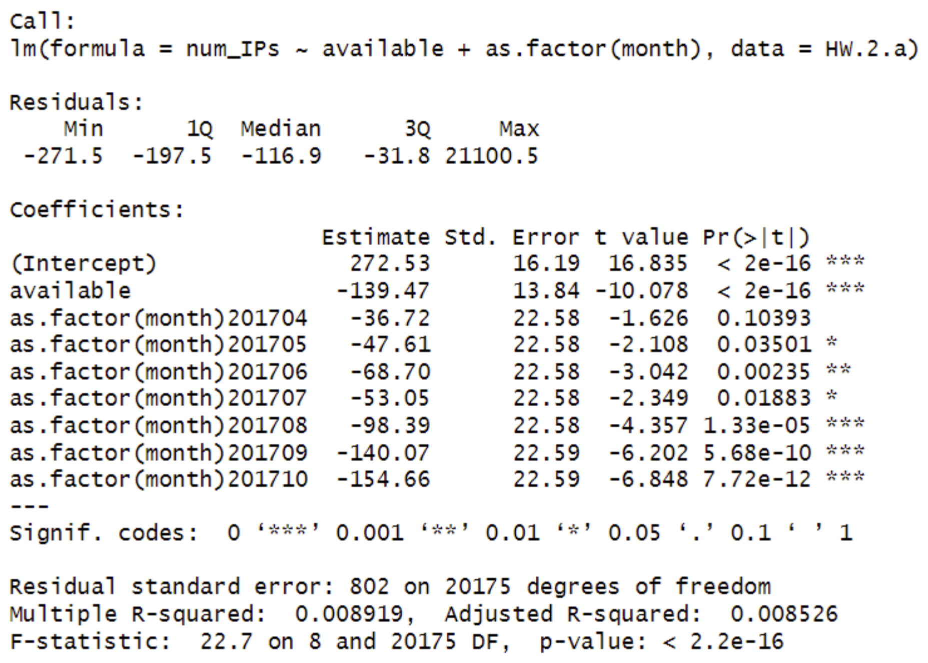


```
> model <- lm(num_IPs ~ available, data = HW.2.a)
> summary(model)

Call:
lm(formula = num_IPs ~ available, data = HW.2.a)

Residuals:
    Min      1Q  Median      3Q     Max 
 -197.1  -190.1  -149.1   -38.3 21174.9 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  198.139      6.367   31.12   <2e-16 ***
available   -141.887     13.860  -10.24   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 803.4 on 20182 degrees of freedom
Multiple R-squared:  0.005166,  Adjusted R-squared:  0.005117 
F-statistic: 104.8 on 1 and 20182 DF,  p-value: < 2.2e-16

> mean(HW.2.a$num_IPs[HW.2.a$available == 1], na.rm = TRUE)
[1] 56.25194
> mean(HW.2.a$num_IPs[HW.2.a$available == 0], na.rm = TRUE)
[1] 198.1393
```

c) The coefficient estimate becomes smaller in magnitude when we include time dummies into the regression equation. The coefficient from part b) overestimated the effect of availability of a movie on streaming platforms on piracy, as it did not account for (omitted) the relationship between time and piracy. There is a time trend in the number of IP addresses illegally exchanging movies. This trend intuitively makes sense for multiple reasons, including improvement of anti-piracy efforts, changes in consumer behavior, etc.

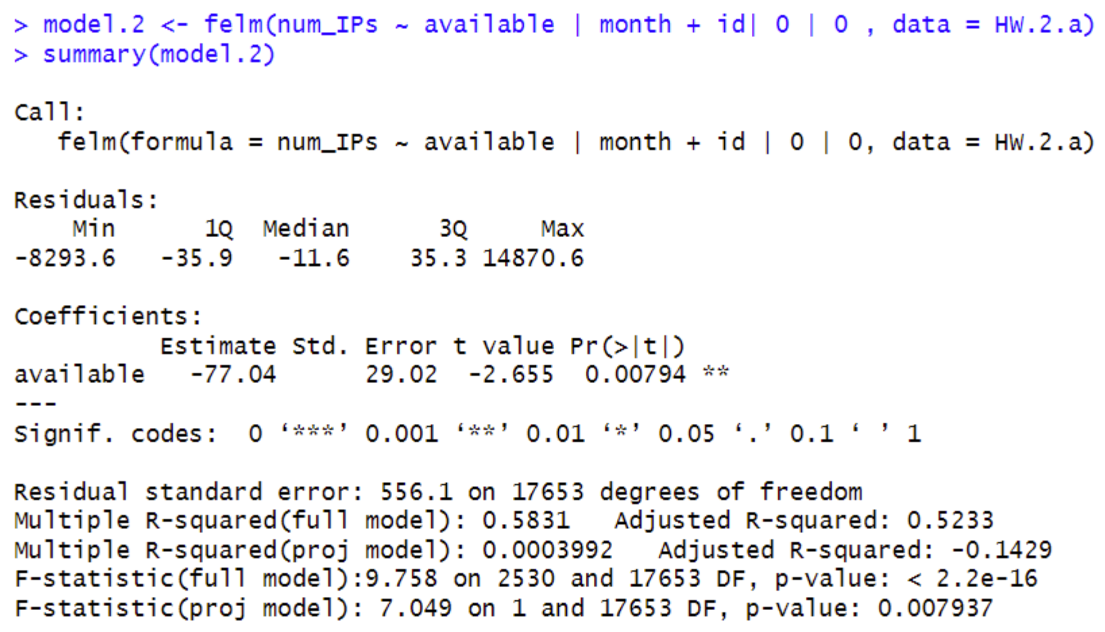


```
> model.1 <- felm(num_IPs ~ available | month | 0 | 0 , data = HW.2.a)
> summary(model.1)

Call:
   felm(formula = num_IPs ~ available | month | 0 | 0, data = HW.2.a)

Residuals:
    Min      1Q  Median      3Q     Max
 -271.5  -197.5  -116.9   -31.8 21100.5

Coefficients:
          Estimate Std. Error t value Pr(>|t|)
available  -139.47      13.84  -10.08   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 802 on 20175 degrees of freedom
Multiple R-squared(full model): 0.008919   Adjusted R-squared: 0.008526
Multiple R-squared(proj model): 0.005009   Adjusted R-squared: 0.004614
F-statistic(full model): 22.7 on 8 and 20175 DF, p-value: < 2.2e-16
F-statistic(proj model): 101.6 on 1 and 20175 DF, p-value: < 2.2e-16
```



```
Call:
lm(formula = num_IPs ~ available + as.factor(month), data = HW.2.a)

Residuals:
    Min      1Q  Median      3Q     Max
 -271.5  -197.5  -116.9   -31.8 21100.5

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)               272.53      16.19  16.835  < 2e-16 ***
available                -139.47      13.84 -10.078  < 2e-16 ***
as.factor(month)201704    -36.72      22.58  -1.626  0.10393
as.factor(month)201705    -47.61      22.58  -2.108  0.03501 *
as.factor(month)201706    -68.70      22.58  -3.042  0.00235 **
as.factor(month)201707    -53.05      22.58  -2.349  0.01883 *
as.factor(month)201708    -98.39      22.58  -4.357 1.33e-05 ***
as.factor(month)201709   -140.07      22.59  -6.202 5.68e-10 ***
as.factor(month)201710   -154.66      22.59  -6.848 7.72e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 802 on 20175 degrees of freedom
Multiple R-squared:  0.008919,  Adjusted R-squared:  0.008526
F-statistic:  22.7 on 8 and 20175 DF,  p-value: < 2.2e-16
```

d) The coefficient estimate has substantially decreased in magnitude after we included time and movie fixed effects into the regression. This implies that the regression coefficients calculated in parts b) overestimated the true relationship between the availability of a movie on streaming platforms and the number of piracy cases, attributing the effect of time and movie-specific differences to the availability factor.

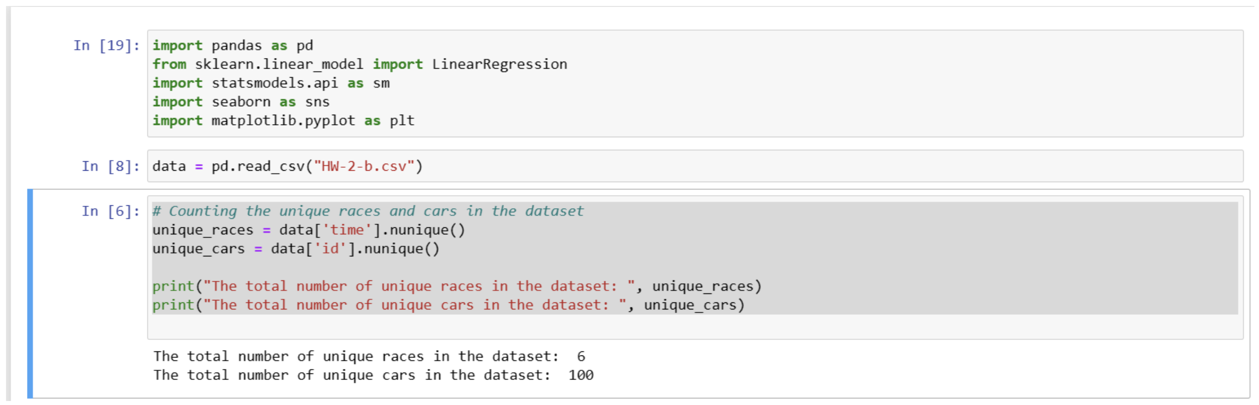


```
> model.2 <- felm(num_IPs ~ available | month + id| 0 | 0 , data = HW.2.a)
> summary(model.2)

Call:
   felm(formula = num_IPs ~ available | month + id | 0 | 0, data = HW.2.a)

Residuals:
    Min      1Q  Median      3Q     Max
-8293.6   -35.9   -11.6    35.3 14870.6

Coefficients:
          Estimate Std. Error t value Pr(>|t|)
available   -77.04      29.02  -2.655  0.00794 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 556.1 on 17653 degrees of freedom
Multiple R-squared(full model): 0.5831   Adjusted R-squared: 0.5233
Multiple R-squared(proj model): 0.0003992   Adjusted R-squared: -0.1429
F-statistic(full model):9.758 on 2530 and 17653 DF, p-value: < 2.2e-16
F-statistic(proj model): 7.049 on 1 and 17653 DF, p-value: 0.007937
```

To conclude, there is a strong (significant at 0.01 level) relationship between the availability and piracy - on average, holding all else equal, availability of a movie on streaming platforms is associated with 77 less IP addresses illegally exchanging that movie. This estimate is considerably different from the one in part b). We believe that the estimate from part d) is more reliable as it is calculated by controlling for time and movie fixed effects.

Question 3:

a) To find number of unique races and cars in the dataset, we write the following code:



```
In [19]: import pandas as pd
         from sklearn.linear_model import LinearRegression
         import statsmodels.api as sm
         import seaborn as sns
         import matplotlib.pyplot as plt

In [8]: data = pd.read_csv("HW-2-b.csv")

In [6]: # Counting the unique races and cars in the dataset
        unique_races = data['time'].nunique()
        unique_cars = data['id'].nunique()

        print("The total number of unique races in the dataset: ", unique_races)
        print("The total number of unique cars in the dataset: ", unique_cars)

        The total number of unique races in the dataset:  6
        The total number of unique cars in the dataset:  100
```

As seen from the screenshot, we can see that there are 6 unique races and 100 unique cars in the dataset.

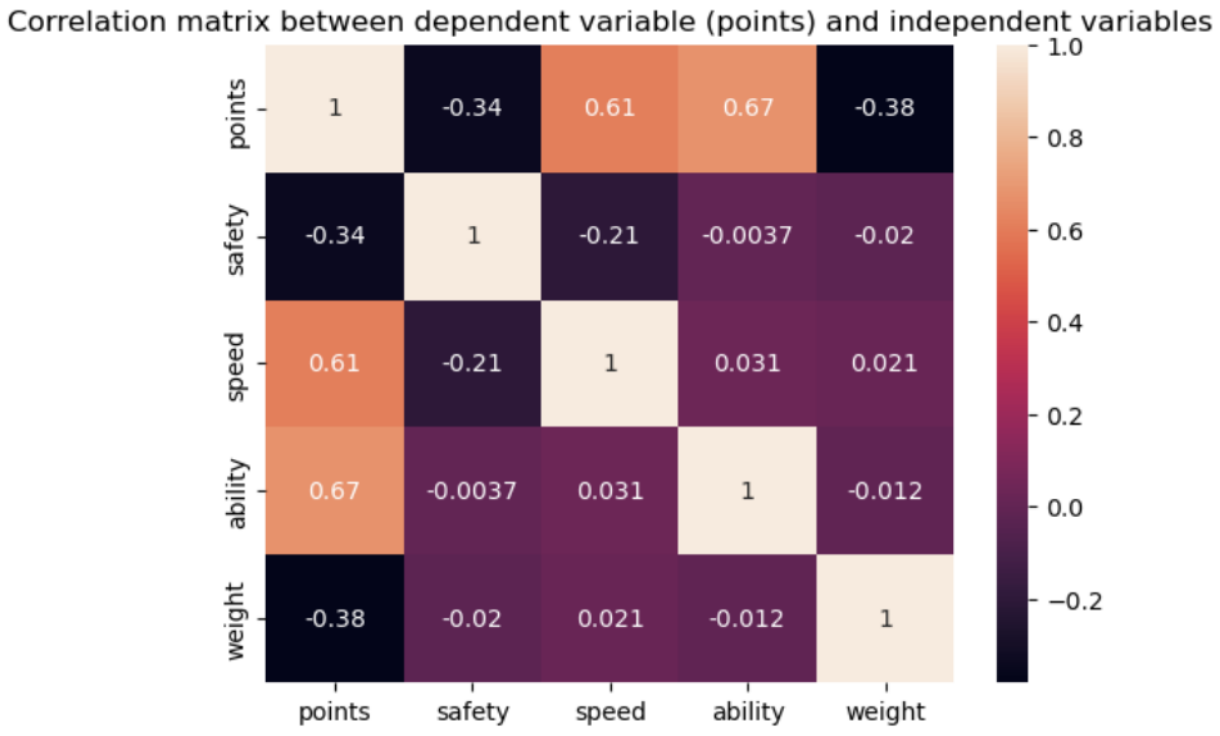
b) To identify the significant relationships between points, safety, speed, ability and weight, we do two steps:

i) Create a correlation matrix to identify the dependency not only between the independent variables and dependent variable, but also between the independent variables themselves

ii) Fit a linear regression model to identify if there is any statistically significant relationships between the dependent variable and the independent variables.

Step 1:

```
# Calculating the correlation matrix for the selected columns
# subsetting data and building a correlation matrix
subset_data = data[['points', 'safety', 'speed', 'ability', 'weight']]
correlation_matrix = subset_data.corr()

# Showing the correlation matrix
correlation_matrix

sns.heatmap(correlation_matrix, annot=True)
plt.title("Correlation matrix between dependent variable (points) and independent variables")
plt.show()
```

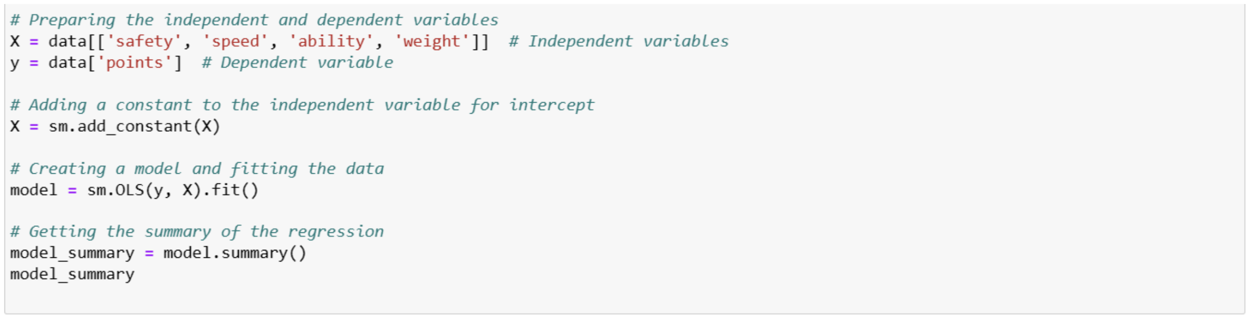


Key observations from the correlation matrix are:

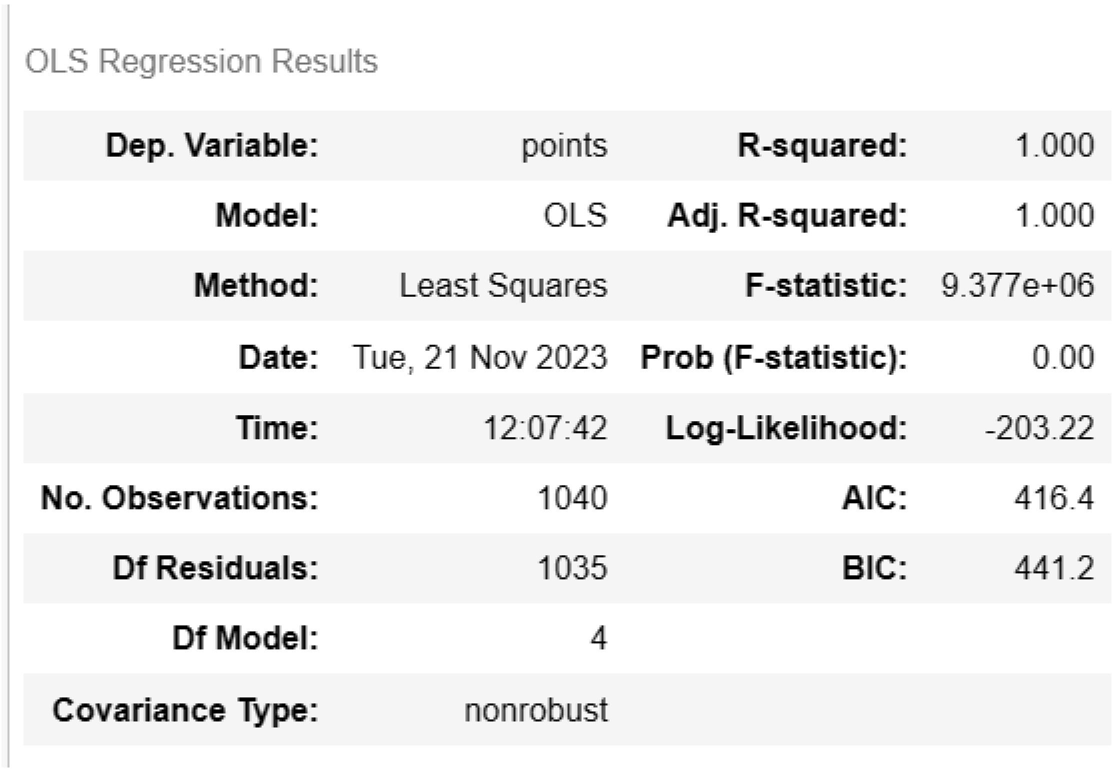
1. Points variable has moderate to strong positive correlations with speed, and ability, but a moderately negative correlation with safety, suggesting that higher ratings in speed and ability are associated with higher points, but higher rating in safety is associated with lower points.
2. Safety and speed ratings show a moderate negative correlation with each other, as higher speeds tend to be less safe for driving.
3. Ability has barely any correlation with both safety and speed ratings (positive in the case of safety and negatively in the case of speed respectively).

4. Weight does not show a significant correlation with other variables, though it has a moderately negative correlation to points.

Step 2: Fitting the linear regression model

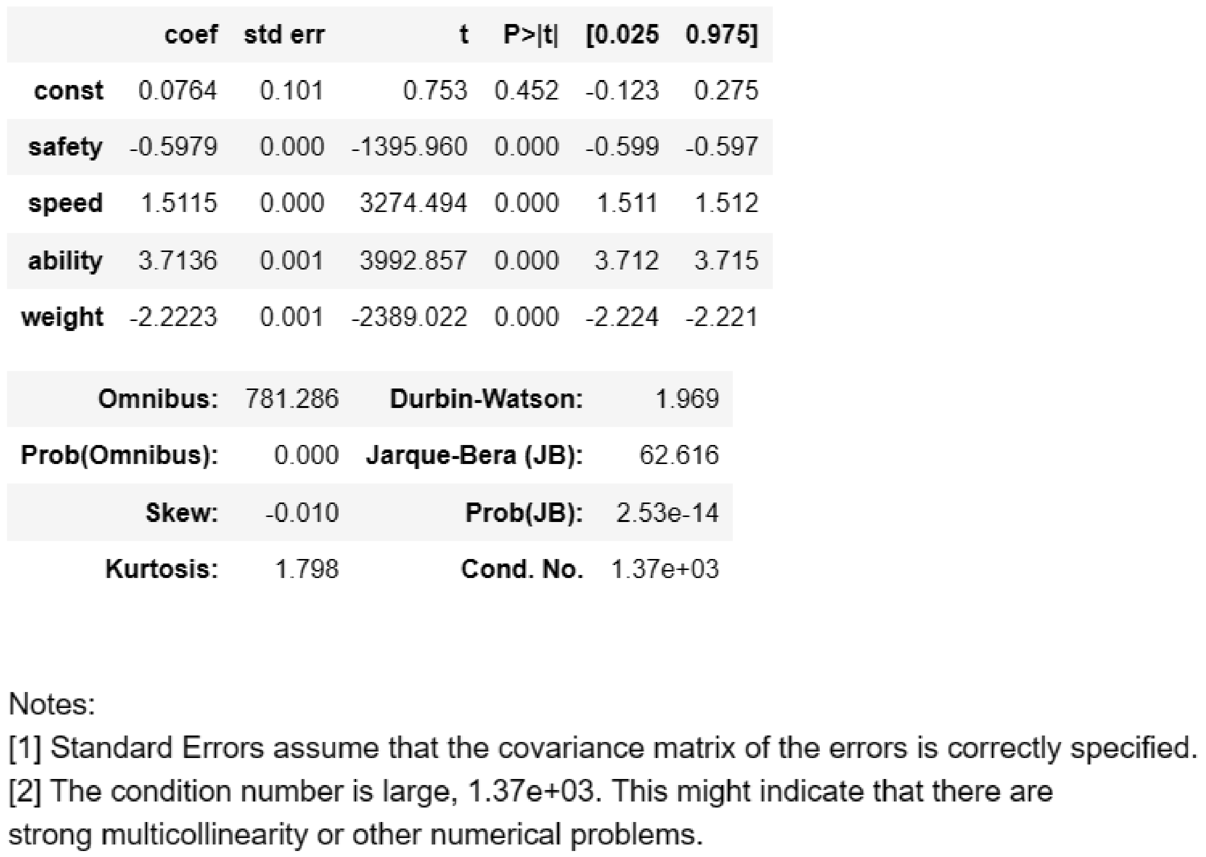


```
# Preparing the independent and dependent variables
X = data[['safety', 'speed', 'ability', 'weight']]  # Independent variables
y = data['points']  # Dependent variable

# Adding a constant to the independent variable for intercept
X = sm.add_constant(X)

# Creating a model and fitting the data
model = sm.OLS(y, X).fit()

# Getting the summary of the regression
model_summary = model.summary()
model_summary
```



OLS Regression Results

Dep. Variable:	points	**R-squared:**	1.000
Model:	OLS	**Adj. R-squared:**	1.000
Method:	Least Squares	**F-statistic:**	9.377e+06
Date:	Tue, 21 Nov 2023	**Prob (F-statistic):**	0.00
Time:	12:07:42	**Log-Likelihood:**	-203.22
No. Observations:	1040	**AIC:**	416.4
Df Residuals:	1035	**BIC:**	441.2
Df Model:	4		
Covariance Type:	nonrobust		



	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0764	0.101	0.753	0.452	-0.123	0.275
safety	-0.5979	0.000	-1395.960	0.000	-0.599	-0.597
speed	1.5115	0.000	3274.494	0.000	1.511	1.512
ability	3.7136	0.001	3992.857	0.000	3.712	3.715
weight	-2.2223	0.001	-2389.022	0.000	-2.224	-2.221

Omnibus:	781.286	**Durbin-Watson:**	1.969
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	62.616
Skew:	-0.010	**Prob(JB):**	2.53e-14
Kurtosis:	1.798	**Cond. No.**	1.37e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.37e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

Observations:

R-squared: The R-squared value is 1.000, indicating that all the variance in the response variable is explained by the independent variables. However, a value of 1.000 is quite unusual and indicates overfitting (or some peculiarity in the data).

Coefficients and their meanings:

Safety: The coefficient for safety is -0.5979, which indicates a negative relationship with points. It means that holding all other variables constant, for each unit increase in safety, the points decrease by about 0.5979 units.

Speed: The coefficient for speed is 1.5115, showing a strong positive relationship. This means holding all other variables constant for each unit increase in speed, the points increase by approximately 1.5115 units.

Ability: The ability has a coefficient of 3.7136, also indicating a strong positive relationship. This means that holding all other variables constant, each unit increase in ability is associated with an increase of about 3.7136 points.

Weight: The coefficient for weight is -2.2223, suggesting a negative relationship. Each unit increase in weight of the driver leads to a decrease of approximately 2.2223 points.

P-values: The P-values for all the coefficients are very close to 0 at 95% confidence level (alpha = 5%), indicating that these relationships are statistically significant.

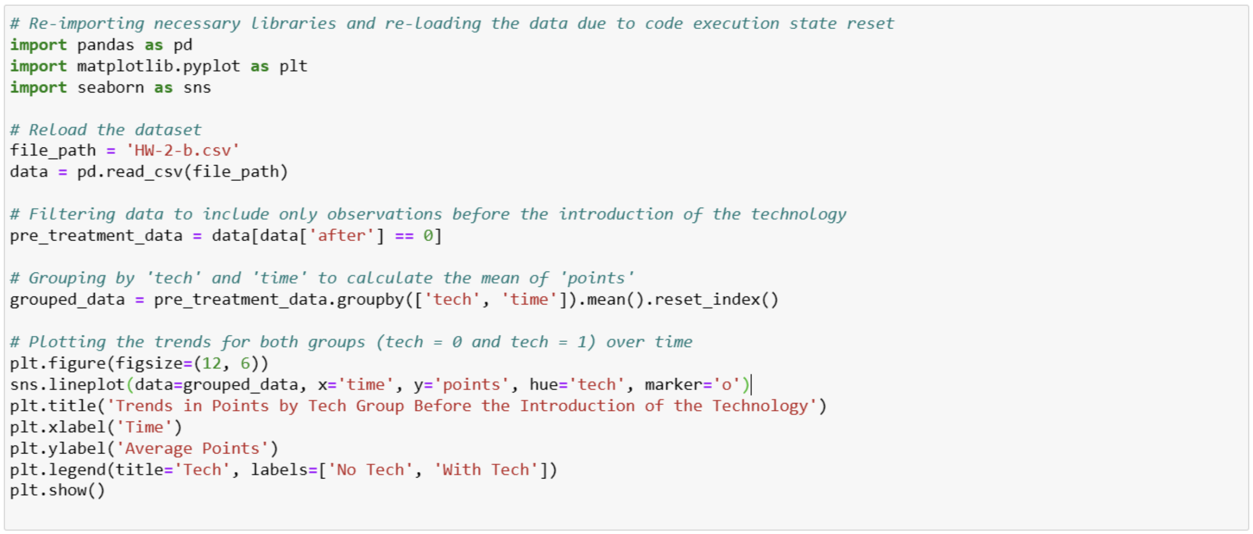
However, there are certain issues with the model.

Issues:

1) From the condition number, we can find that there is large multicollinearity in the model and hence, we can conclude that we have unstable coefficients meaning a small change in the data will lead to large changes in the coefficients.
2) Overall, given our R^2 value indicating the presence of overfitting and multicollinearity, our model may be unreliable in terms of predictive ability to unseen data as the individual predictor variables may not be reliable.

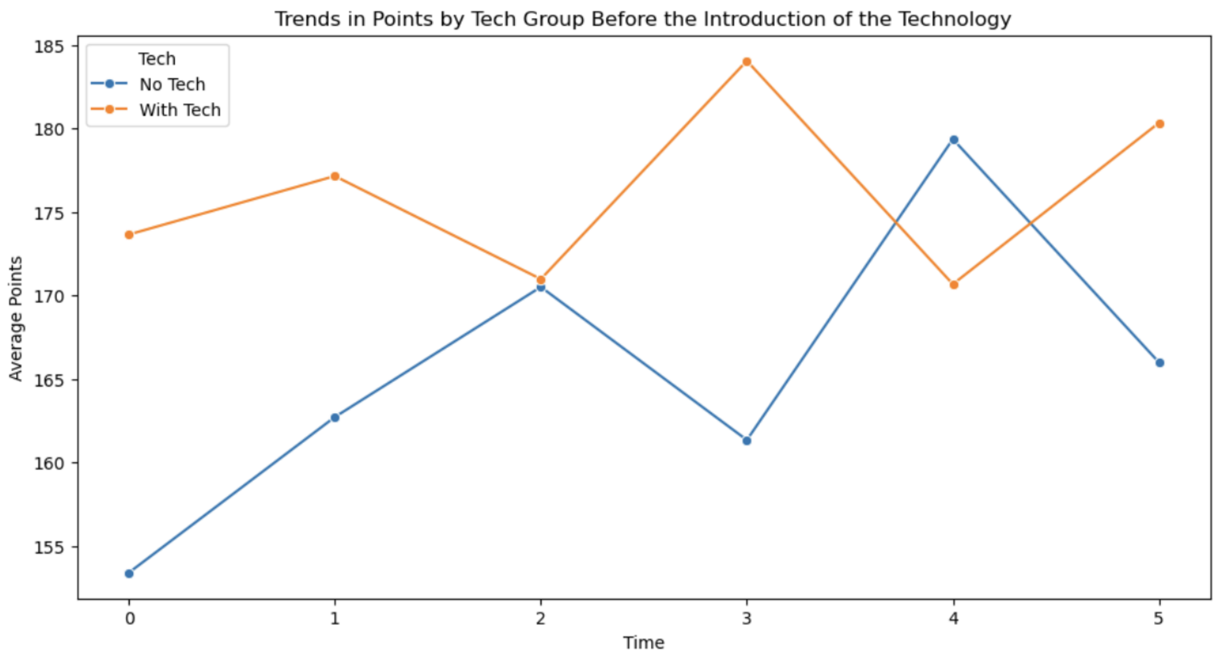
c)
To estimate the effects of the impact of technology on the points, safety and ability, we would need to implement the difference in differences regression model.

First, we need to test the parallel trends assumption. The parallel trends assumption is tested as follows:



```python
# Re-importing necessary libraries and re-loading the data due to code execution state reset
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Reload the dataset
file_path = 'HW-2-b.csv'
data = pd.read_csv(file_path)

# Filtering data to include only observations before the introduction of the technology
pre_treatment_data = data[data['after'] == 0]

# Grouping by 'tech' and 'time' to calculate the mean of 'points'
grouped_data = pre_treatment_data.groupby(['tech', 'time']).mean().reset_index()

# Plotting the trends for both groups (tech = 0 and tech = 1) over time
plt.figure(figsize=(12, 6))
sns.lineplot(data=grouped_data, x='time', y='points', hue='tech', marker='o')
plt.title('Trends in Points by Tech Group Before the Introduction of the Technology')
plt.xlabel('Time')
plt.ylabel('Average Points')
plt.legend(title='Tech', labels=['No Tech', 'With Tech'])
plt.show()
```

The parallel trends plot:



From the plot we can infer that the lines are roughly parallel, which suggests that the trends in points for both groups were similar before the introduction of the technology. This supports the parallel trends assumption, a critical requirement for the Difference-in-Differences approach.

Now we would need to match the cars with similar values for speed, safety, ability and weight, given they have a significant effect on points from the previous regression that we observed.

We achieve it through the following code:

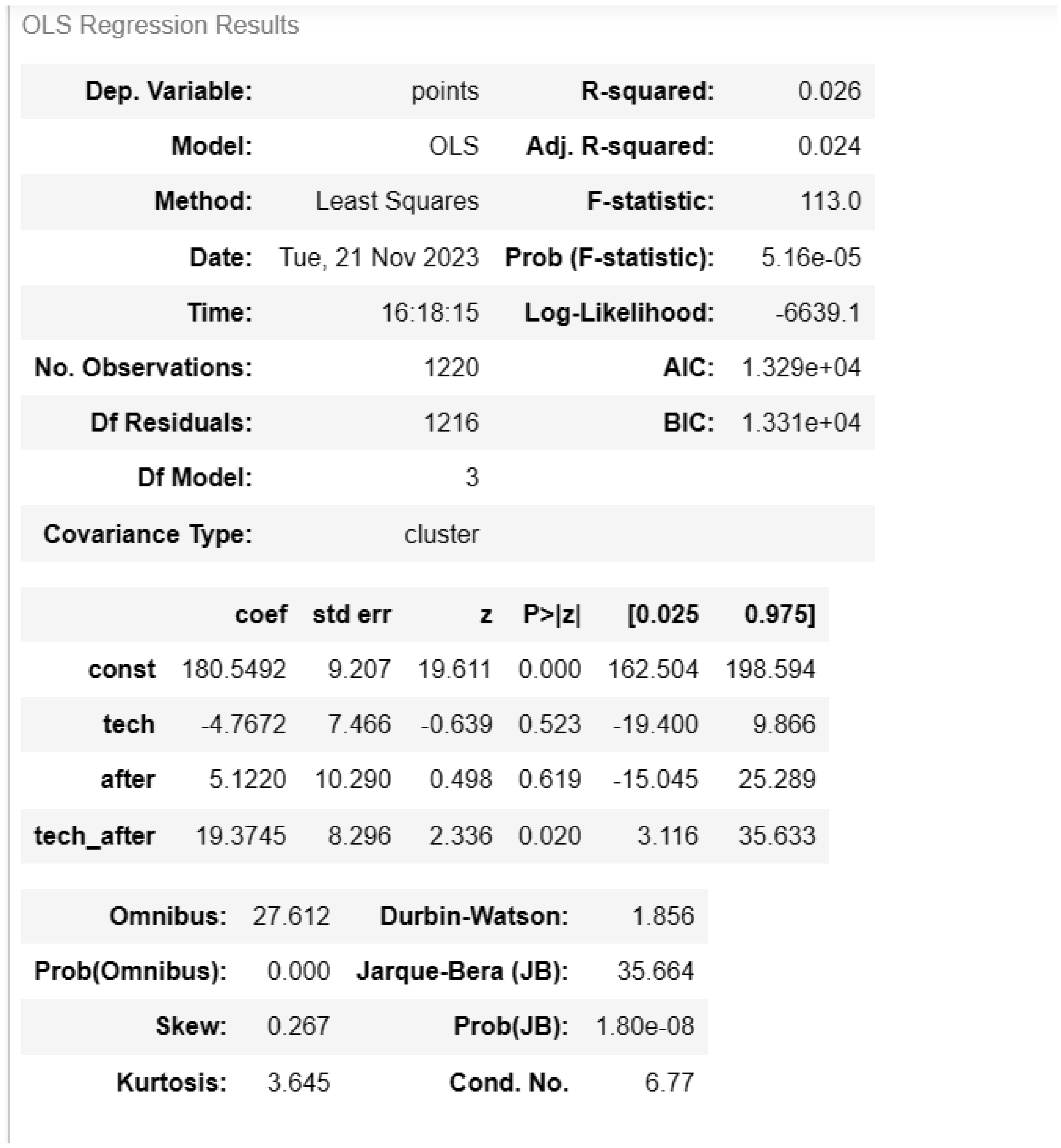
```
# Step 1: Propensity Score Matching

# Logistic regression model to estimate propensity scores
# Using 'tech' as the dependent variable and other covariates for matching
logit_model = LogisticRegression()
X_match = data[['safety', 'speed', 'weight', 'ability']]
y_match = data['tech']

# Fitting the logistic regression model
logit_model.fit(X_match, y_match)

# Estimating propensity scores
propensity_scores = logit_model.predict_proba(X_match)[:, 1]

# Adding the propensity scores to the dataset
data['propensity_score'] = propensity_scores

# Implementing matching - Nearest Neighbor Matching
# For simplicity, we take a 1-to-1 matching without replacement

# Separating treatment and control groups
treatment = data[data['tech'] == 1]
control = data[data['tech'] == 0]

# Calculating pairwise distance matrix between treatment and control groups based on propensity scores
distance_matrix = pairwise_distances(treatment['propensity_score'].values.reshape(-1, 1),
                                     control['propensity_score'].values.reshape(-1, 1))

# For each treatment unit, find the index of the closest control unit
min_distances_indices = np.argmin(distance_matrix, axis=1)
```

```python
# Separating treatment and control groups
treatment = data[data['tech'] == 1]
control = data[data['tech'] == 0]

# Calculating pairwise distance matrix between treatment and control groups based on propensity scores
distance_matrix = pairwise_distances(treatment['propensity_score'].values.reshape(-1, 1),
                                     control['propensity_score'].values.reshape(-1, 1))

# For each treatment unit, find the index of the closest control unit
min_distances_indices = np.argmin(distance_matrix, axis=1)

# Extracting the matched control units
matched_control = control.iloc[min_distances_indices]

# Concatenating the matched treatment and control units
matched_data = pd.concat([treatment, matched_control])

matched_data.head()  # Displaying the first few rows of the matched data
```

	id	tech	after	time	safety	speed	weight	ability	points	propensity_score
1	2	1	0	0	68.452895	58.336126	46.238298	83.298663	254	0.522074
2	3	1	0	0	9.825860	63.345852	78.790024	82.463555	221	0.771961
4	5	1	0	0	14.739271	52.761438	48.631813	81.247646	265	0.742687
6	7	1	0	0	24.823326	31.513777	80.874570	81.561710	156	0.657966
7	8	1	0	0	3.956918	60.637896	68.608305	76.379461	221	0.772849

Then we build a model with points as dependent variable, tech, after and their interaction as predictors with id used to calculate cluster standard errors.

```python
# Preparing the data for DiD regression with the matched dataset
# Creating the interaction term between 'tech' and 'after' for the DiD analysis
matched_data['tech_after'] = matched_data['tech'] * matched_data['after']

# Independent variables: 'tech', 'after', and the interaction term 'tech_after'
X_matched = matched_data[['tech', 'after', 'tech_after']]

# Adding a constant to the independent variables
X_matched = sm.add_constant(X_matched)

# Dependent variable: 'points'
y_matched = matched_data['points']

# Creating the DiD regression model with matched data
did_model_matched = sm.OLS(y_matched, X_matched)

# Fitting the model with clustered standard errors by 'time'
clustered_results_matched = did_model_matched.fit(cov_type='cluster', cov_kwds={'groups': matched_data['id']})

# Getting the summary of the model with clustered standard errors by time
clustered_results_matched_summary = clustered_results_matched.summary()
clustered_results_matched_summary
```



OLS Regression Results

Dep. Variable:	points	**R-squared:**	0.026
Model:	OLS	**Adj. R-squared:**	0.024
Method:	Least Squares	**F-statistic:**	113.0
Date:	Tue, 21 Nov 2023	**Prob (F-statistic):**	5.16e-05
Time:	16:18:15	**Log-Likelihood:**	-6639.1
No. Observations:	1220	**AIC:**	1.329e+04
Df Residuals:	1216	**BIC:**	1.331e+04
Df Model:	3		
Covariance Type:	cluster		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	180.5492	9.207	19.611	0.000	162.504	198.594
tech	-4.7672	7.466	-0.639	0.523	-19.400	9.866
after	5.1220	10.290	0.498	0.619	-15.045	25.289
tech_after	19.3745	8.296	2.336	0.020	3.116	35.633

Omnibus:	27.612	**Durbin-Watson:**	1.856
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	35.664
Skew:	0.267	**Prob(JB):**	1.80e-08
Kurtosis:	3.645	**Cond. No.**	6.77

Coefficients:
Constant (Baseline): The constant term is 180.5492, highly significant, indicating the baseline points when both tech and after are 0.
tech: Coefficient = -4.7672, p-value = 0.523, suggesting no significant effect of the technology alone on points.
after: Coefficient = 5.1220, p-value = 0.619, indicating no significant effect of the time period after the introduction of the technology.

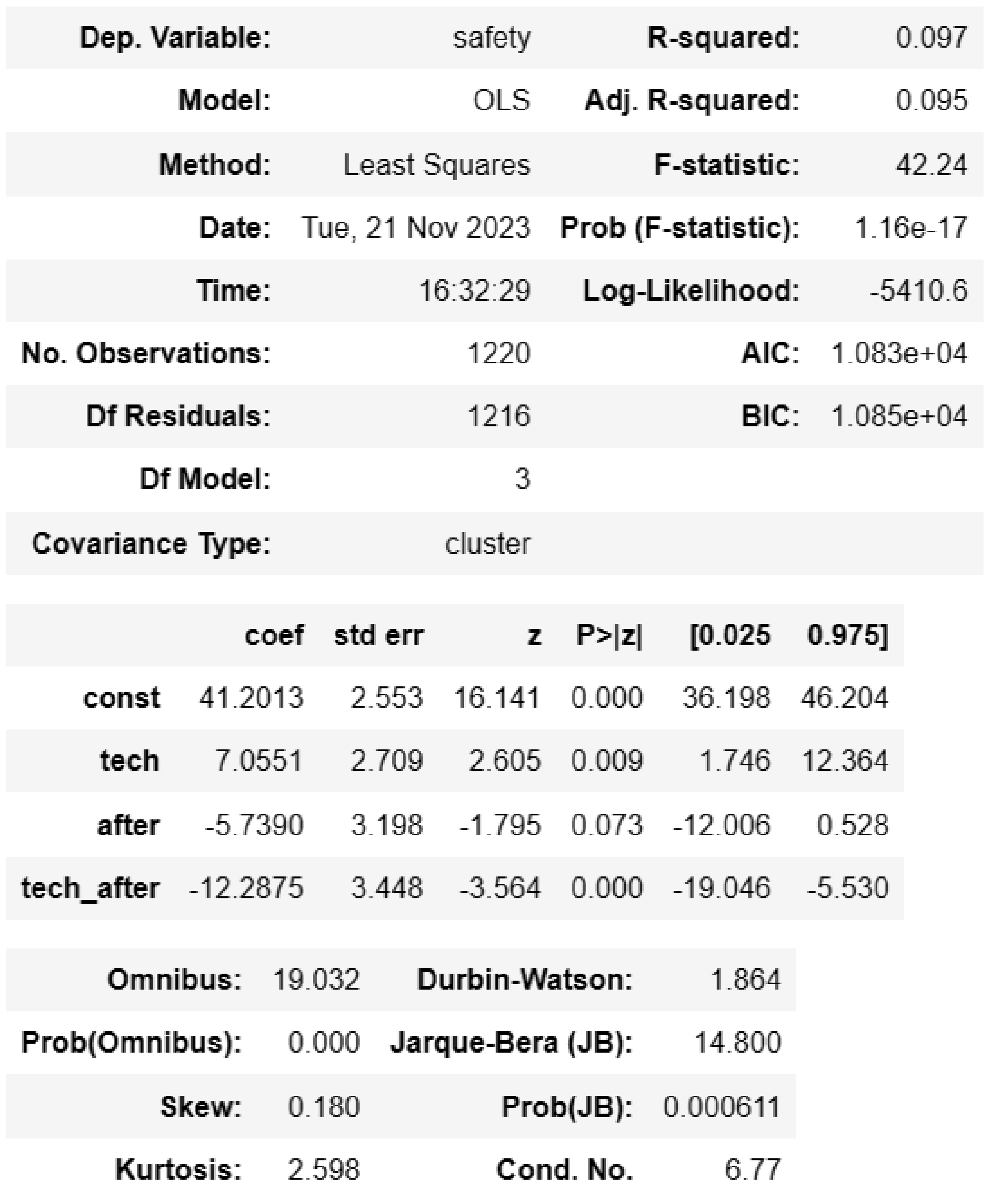
tech_after (DiD Estimate): Coefficient = 19.3745, p-value = 0.020, indicating a positive interaction effect, which is statistically significant at the conventional 0.05 level.
Clustering standard errors by id accounts for potential within-unit (car) correlation, leading to potentially more accurate standard error estimates. The id-clustered standard errors slightly alter the significance of some coefficients compared to the non-clustered model.

These results suggest that while there is a positive effect of the technology on points, the effect is statistically significant when accounting for clustering by id. The validity of the causal interpretation still depends on whether the assumptions of DiD analysis, particularly the parallel trends assumption and the quality of the matching process, are met

DiD Model for safety:

```
# Step 1: Propensity Score Matching for the new setup
# Using 'speed', 'weight', 'ability', and 'points' for matching

# Fitting the logistic regression model for propensity score estimation
logit_model.fit(X_match, y_match)  # X_match and y_match are already defined with the new matching covariates

# Estimating propensity scores with the new model
propensity_scores_new = logit_model.predict_proba(X_match)[:, 1]

# Adding the new propensity scores to the dataset
data['propensity_score_new'] = propensity_scores_new

# Implementing matching - Nearest Neighbor Matching for the new setup

# Separating treatment and control groups
treatment_new = data[data['tech'] == 1]
control_new = data[data['tech'] == 0]

# Calculating pairwise distance matrix based on the new propensity scores
distance_matrix_new = pairwise_distances(treatment_new['propensity_score_new'].values.reshape(-1, 1),
                                         control_new['propensity_score_new'].values.reshape(-1, 1))

# Finding the index of the closest control unit for each treatment unit
min_distances_indices_new = np.argmin(distance_matrix_new, axis=1)

# Extracting the matched control units
matched_control_new = control_new.iloc[min_distances_indices_new]
```

```python
# Finding the index of the closest control unit for each treatment unit
min_distances_indices_new = np.argmin(distance_matrix_new, axis=1)

# Extracting the matched control units
matched_control_new = control_new.iloc[min_distances_indices_new]

# Concatenating the matched treatment and control units
matched_data_new = pd.concat([treatment_new, matched_control_new])

# Step 2: DiD Regression with the matched data, considering 'safety' as the dependent variable

# Creating the interaction term 'tech_after' for the DiD analysis
matched_data_new['tech_after'] = matched_data_new['tech'] * matched_data_new['after']

# Independent variables: 'tech', 'after', and 'tech_after'
X_matched_safety = matched_data_new[['tech', 'after', 'tech_after']]
X_matched_safety = sm.add_constant(X_matched_safety)

# Dependent variable: 'safety'
y_matched_safety = matched_data_new['safety']

# Creating the DiD regression model with matched data
did_model_matched_safety = sm.OLS(y_matched_safety, X_matched_safety)

# Fitting the model with clustered standard errors by 'id'
clustered_results_matched_safety = did_model_matched_safety.fit(cov_type='cluster', cov_kwds={'groups': matched_data_new['id']})

# Getting the summary of the model
clustered_results_matched_safety_summary = clustered_results_matched_safety.summary()
clustered_results_matched_safety_summary
```



Dep. Variable:	safety	**R-squared:**	0.097
Model:	OLS	**Adj. R-squared:**	0.095
Method:	Least Squares	**F-statistic:**	42.24
Date:	Tue, 21 Nov 2023	**Prob (F-statistic):**	1.16e-17
Time:	16:32:29	**Log-Likelihood:**	-5410.6
No. Observations:	1220	**AIC:**	1.083e+04
Df Residuals:	1216	**BIC:**	1.085e+04
Df Model:	3		
Covariance Type:	cluster		

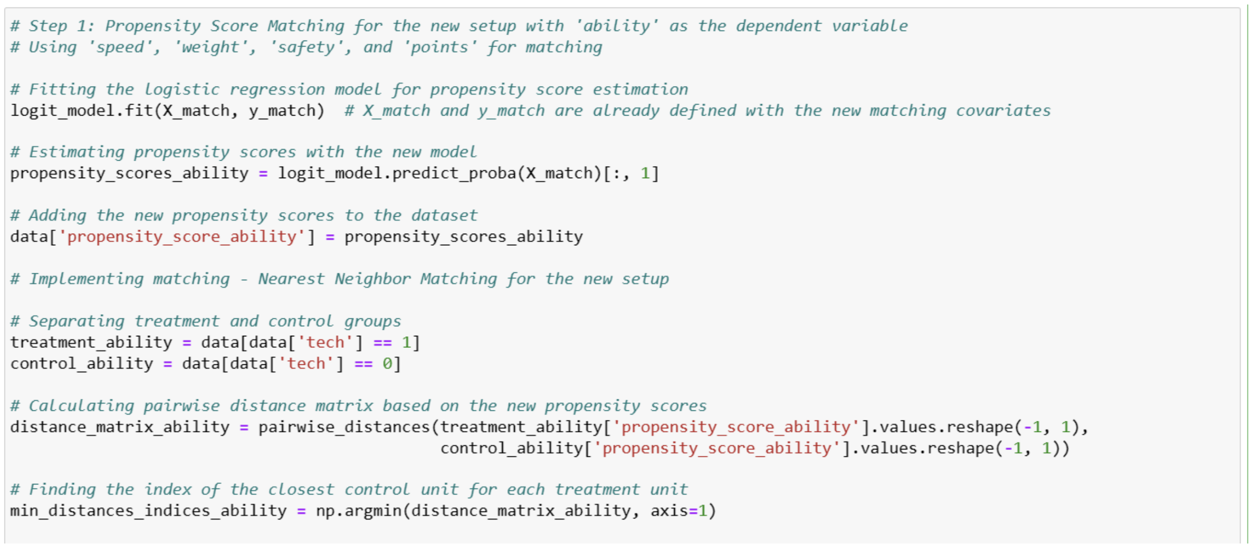
	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	41.2013	2.553	16.141	0.000	36.198	46.204
tech	7.0551	2.709	2.605	0.009	1.746	12.364
after	-5.7390	3.198	-1.795	0.073	-12.006	0.528
tech_after	-12.2875	3.448	-3.564	0.000	-19.046	-5.530

Omnibus:	19.032	**Durbin-Watson:**	1.864
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	14.800
Skew:	0.180	**Prob(JB):**	0.000611
Kurtosis:	2.598	**Cond. No.**	6.77

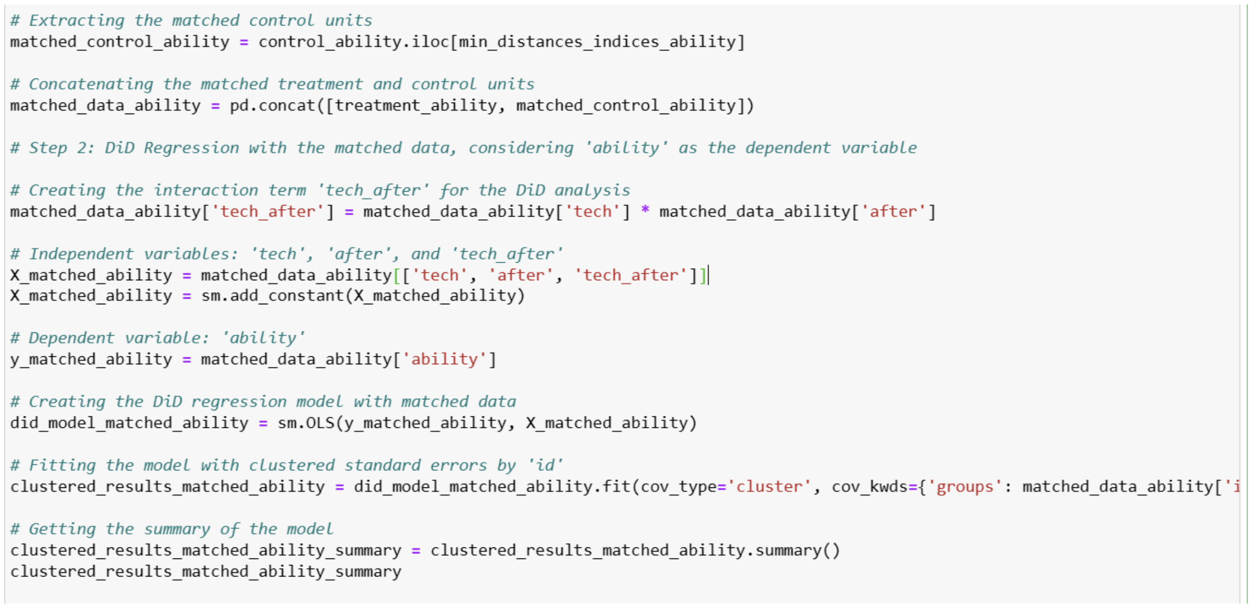
Coefficients:
Constant (Baseline): The constant term is 41.2013, highly significant, indicating the baseline safety rating when both tech and after are 0.
tech: Coefficient = 7.0551, p-value = 0.009, suggesting a significant effect of the technology alone on safety.
after: Coefficient = -5.7390, p-value = 0.073, indicating a potential negative effect of the time period after the introduction of the technology on safety, but it's not statistically significant at the conventional 0.05 level.
tech_after (DiD Estimate): Coefficient = -12.2875, p-value = 0.000, indicating a significant and negative interaction effect.

The significant and negative tech_after coefficient suggests that the introduction of the technology had a negative effect on safety, even after controlling for similarity in speed, weight, ability, and points through matching and clustering standard errors by id. The results suggest a potential trade-off between the adoption of the technology and safety, as measured in this model.

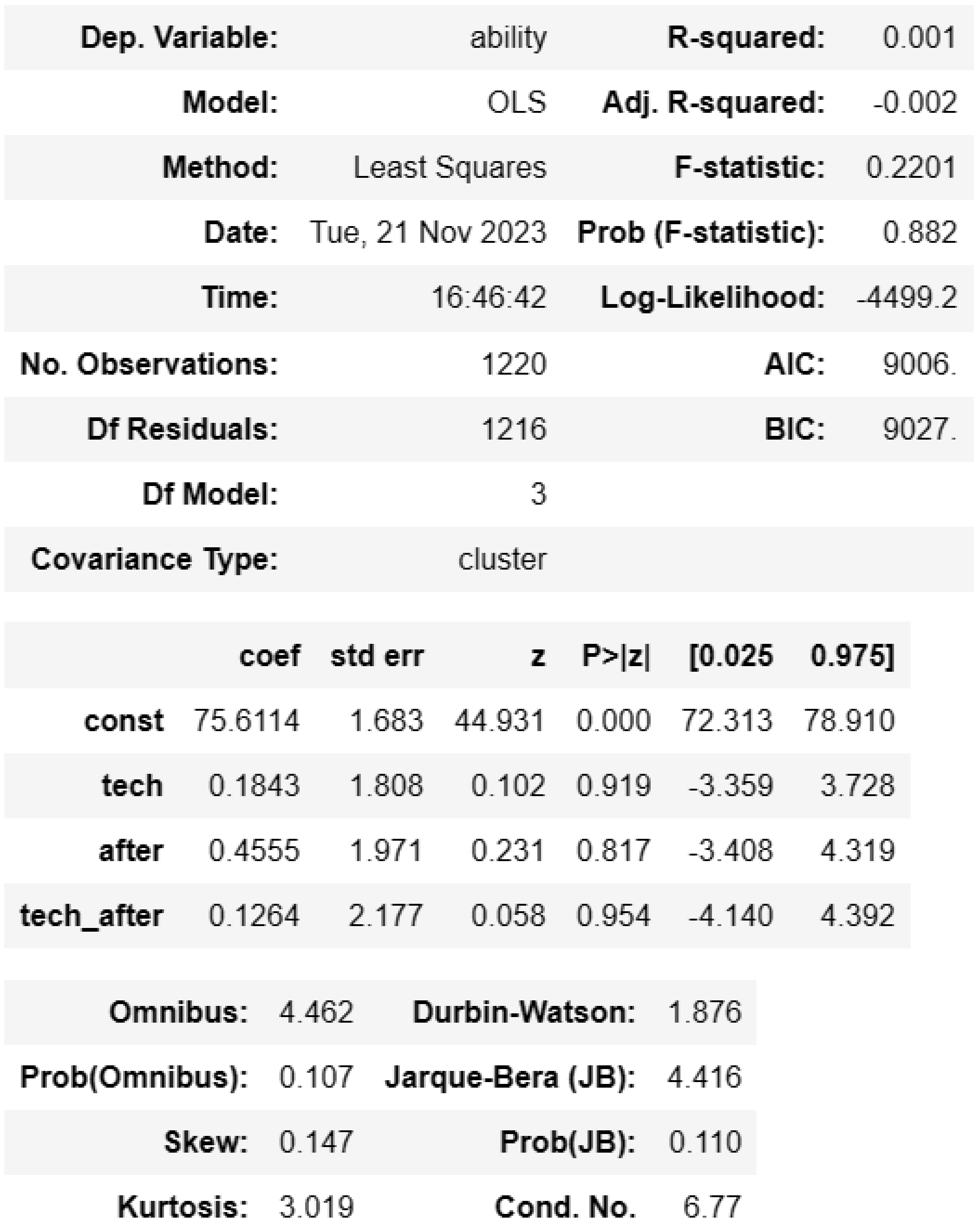
DiD model for ability:



```
# Step 1: Propensity Score Matching for the new setup with 'ability' as the dependent variable
# Using 'speed', 'weight', 'safety', and 'points' for matching

# Fitting the logistic regression model for propensity score estimation
logit_model.fit(X_match, y_match)  # X_match and y_match are already defined with the new matching covariates

# Estimating propensity scores with the new model
propensity_scores_ability = logit_model.predict_proba(X_match)[:, 1]

# Adding the new propensity scores to the dataset
data['propensity_score_ability'] = propensity_scores_ability

# Implementing matching - Nearest Neighbor Matching for the new setup

# Separating treatment and control groups
treatment_ability = data[data['tech'] == 1]
control_ability = data[data['tech'] == 0]

# Calculating pairwise distance matrix based on the new propensity scores
distance_matrix_ability = pairwise_distances(treatment_ability['propensity_score_ability'].values.reshape(-1, 1),
                                             control_ability['propensity_score_ability'].values.reshape(-1, 1))

# Finding the index of the closest control unit for each treatment unit
min_distances_indices_ability = np.argmin(distance_matrix_ability, axis=1)
```



```
# Extracting the matched control units
matched_control_ability = control_ability.iloc[min_distances_indices_ability]

# Concatenating the matched treatment and control units
matched_data_ability = pd.concat([treatment_ability, matched_control_ability])

# Step 2: DiD Regression with the matched data, considering 'ability' as the dependent variable

# Creating the interaction term 'tech_after' for the DiD analysis
matched_data_ability['tech_after'] = matched_data_ability['tech'] * matched_data_ability['after']

# Independent variables: 'tech', 'after', and 'tech_after'
X_matched_ability = matched_data_ability[['tech', 'after', 'tech_after']]
X_matched_ability = sm.add_constant(X_matched_ability)

# Dependent variable: 'ability'
y_matched_ability = matched_data_ability['ability']

# Creating the DiD regression model with matched data
did_model_matched_ability = sm.OLS(y_matched_ability, X_matched_ability)

# Fitting the model with clustered standard errors by 'id'
clustered_results_matched_ability = did_model_matched_ability.fit(cov_type='cluster', cov_kwds={'groups': matched_data_ability['i

# Getting the summary of the model
clustered_results_matched_ability_summary = clustered_results_matched_ability.summary()
clustered_results_matched_ability_summary
```



Dep. Variable:	ability	**R-squared:**	0.001
Model:	OLS	**Adj. R-squared:**	-0.002
Method:	Least Squares	**F-statistic:**	0.2201
Date:	Tue, 21 Nov 2023	**Prob (F-statistic):**	0.882
Time:	16:46:42	**Log-Likelihood:**	-4499.2
No. Observations:	1220	**AIC:**	9006.
Df Residuals:	1216	**BIC:**	9027.
Df Model:	3		
Covariance Type:	cluster		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	75.6114	1.683	44.931	0.000	72.313	78.910
tech	0.1843	1.808	0.102	0.919	-3.359	3.728
after	0.4555	1.971	0.231	0.817	-3.408	4.319
tech_after	0.1264	2.177	0.058	0.954	-4.140	4.392

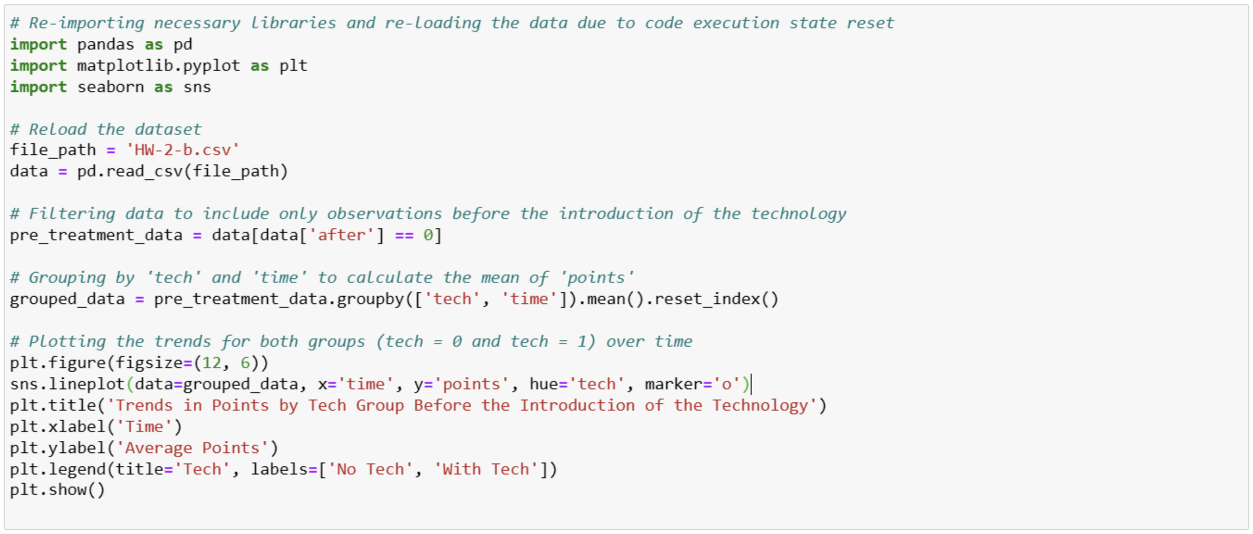
Omnibus:	4.462	**Durbin-Watson:**	1.876
Prob(Omnibus):	0.107	**Jarque-Bera (JB):**	4.416
Skew:	0.147	**Prob(JB):**	0.110
Kurtosis:	3.019	**Cond. No.**	6.77

Coefficients:
Constant (Baseline): The constant term is 75.6114, highly significant, indicating the baseline ability rating when both tech and after are 0.
tech: Coefficient = 0.1843, p-value = 0.919, suggesting no significant effect of the technology alone on ability.
after: Coefficient = 0.4555, p-value = 0.817, indicating no significant effect of the time period after the introduction of the technology on ability.

tech_after (DiD Estimate): Coefficient = 0.1264, p-value = 0.954, indicating no significant interaction effect.

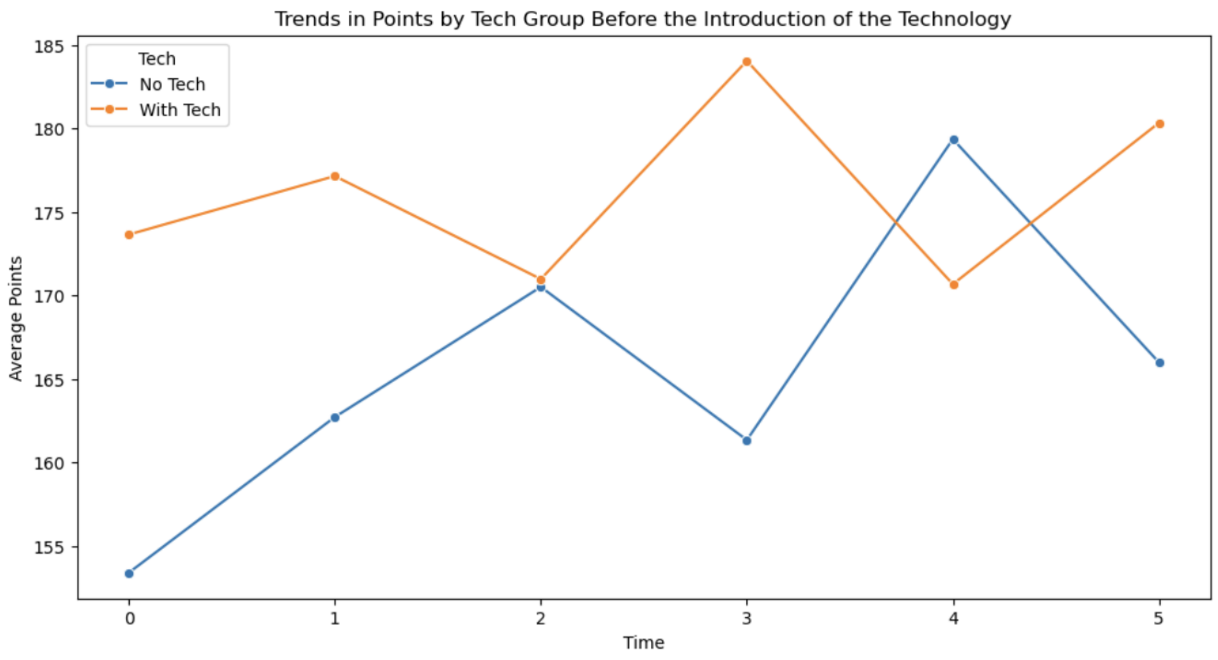
The results suggest that the introduction of the technology does not have a significant effect on the ability of the car drivers, as measured in this model. The low R-squared value and the lack of significance in the tech_after coefficient suggest that the variables included in the model do not explain much of the variation in ability.

d)
First, for the points model, we need to test the parallel trends assumption. The parallel trends assumption is tested as follows:



```python
# Re-importing necessary libraries and re-loading the data due to code execution state reset
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Reload the dataset
file_path = 'HW-2-b.csv'
data = pd.read_csv(file_path)

# Filtering data to include only observations before the introduction of the technology
pre_treatment_data = data[data['after'] == 0]

# Grouping by 'tech' and 'time' to calculate the mean of 'points'
grouped_data = pre_treatment_data.groupby(['tech', 'time']).mean().reset_index()

# Plotting the trends for both groups (tech = 0 and tech = 1) over time
plt.figure(figsize=(12, 6))
sns.lineplot(data=grouped_data, x='time', y='points', hue='tech', marker='o')
plt.title('Trends in Points by Tech Group Before the Introduction of the Technology')
plt.xlabel('Time')
plt.ylabel('Average Points')
plt.legend(title='Tech', labels=['No Tech', 'With Tech'])
plt.show()
```

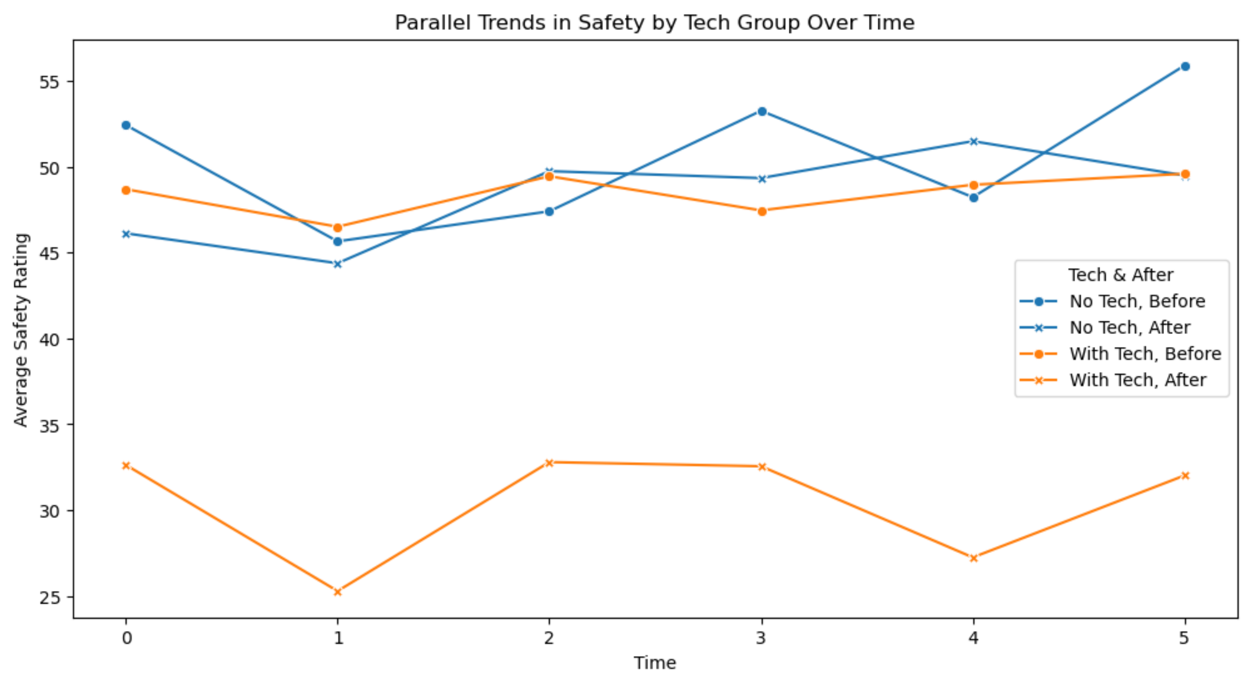
The parallel trends plot:



From the plot we can infer that the lines are roughly parallel, which suggests that the trends in points for both groups were similar before the introduction of the technology. This supports the parallel trends assumption, a critical requirement for the Difference-in-Differences approach.

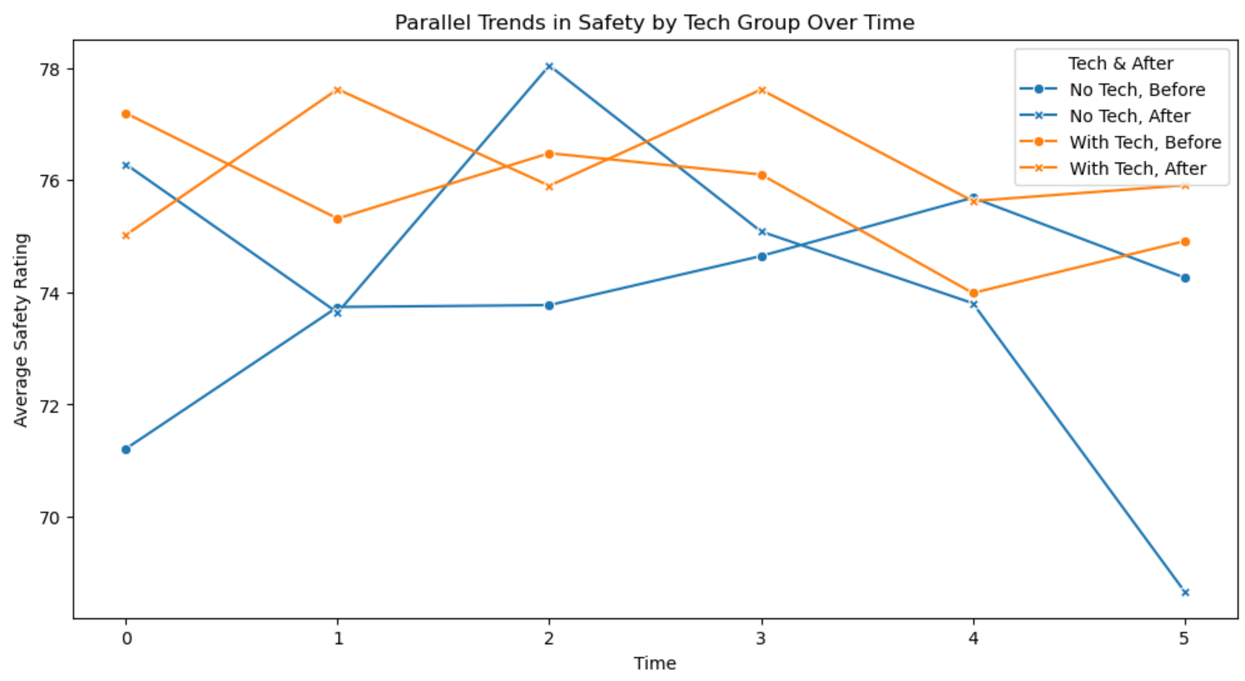
Parallel trends assumption test for the safety model:

```
# Preparing the data for visualizing parallel trends with 'safety' as the dependent variable
# Filtering data to include observations for both before and after the introduction of the technology
grouped_data_safety = data.groupby(['tech', 'after', 'time']).mean()['safety'].reset_index()

# Plotting the trends for both groups (tech = 0 and tech = 1) over time for 'safety'
plt.figure(figsize=(12, 6))
sns.lineplot(data=grouped_data_safety, x='time', y='safety', hue='tech', style='after', markers=True, dashes=False)
plt.title('Parallel Trends in Safety by Tech Group Over Time')
plt.xlabel('Time')
plt.ylabel('Average Safety Rating')
plt.legend(title='Tech & After', labels=['No Tech, Before', 'No Tech, After', 'With Tech, Before', 'With Tech, After'])
plt.show()
```



The trends look roughly parallel and hence reflect the causal estimate given the interaction in part c for the model

```python
# Preparing the data for visualizing parallel trends with 'safety' as the dependent variable
# Filtering data to include observations for both before and after the introduction of the technology
grouped_data_safety = data.groupby(['tech', 'after', 'time']).mean()['ability'].reset_index()

# Plotting the trends for both groups (tech = 0 and tech = 1) over time for 'safety'
plt.figure(figsize=(12, 6))
sns.lineplot(data=grouped_data_safety, x='time', y='ability', hue='tech', style='after', markers=True, dashes=False)
plt.title('Parallel Trends in Safety by Tech Group Over Time')
plt.xlabel('Time')
plt.ylabel('Average Safety Rating')
plt.legend(title='Tech & After', labels=['No Tech, Before', 'No Tech, After', 'With Tech, Before', 'With Tech, After'])
plt.show()
```



The parallel trends assumption is not met here and the effect is not statistically significant. Hence , it is not causal.