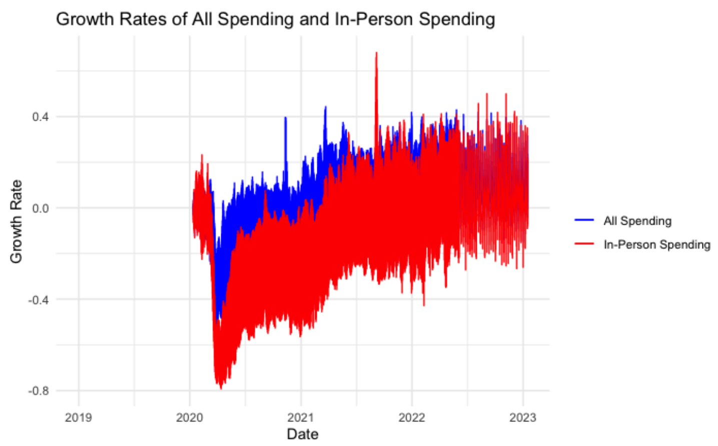
Part 1

1. Summary Statistics & Plot

```{r}
# Load required libraries
library(dplyr)
library(ggplot2)

# Load the Opportunity insights spending data
OI_spend_city <- read.delim2("OI_spend_city.txt", header = TRUE, sep =",", dec = ".")

# Generate a "date" variable from year, month, day
OI_spend_city$date <- as.Date(with(OI_spend_city, paste(year, month, day, sep="-")), "%Y-%m-%d")

tail(OI_spend_city)

# Transforming growth rate columns to numeric
OI_spend_city$spend_all <- as.numeric(OI_spend_city$spend_all)
OI_spend_city$spend_inperson <- as.numeric(OI_spend_city$spend_inperson)

# Report summary statistics for the growth rates of all spending and in-person spending
summary(OI_spend_city$spend_all)


summary(OI_spend_city$spend_inperson)

# Plot the growth rates of all spending and in-person spending

ggplot(OI_spend_city, aes(x=date)) +
  geom_line(aes(y=spend_all, colour="All Spending")) +
  geom_line(aes(y=spend_inperson, colour="In-Person Spending")) +
  labs(title="Growth Rates of All Spending and In-Person Spending",
       x="Date", y="Growth Rate") +
  theme_minimal() +
  scale_colour_manual("",
                      breaks = c("All Spending", "In-Person Spending"),
                      values = c("All Spending"="blue", "In-Person Spending"="red"))
```

```
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
-0.5480 -0.0638  0.0181  0.0052  0.0896  0.4430    2477

Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
-0.7950 -0.2960 -0.1200 -0.1565  0.0016  0.6800    2477
```



2. Keep only from January 2019 to June 2022.

~~~
```{r}
# Add a column that shows "yyyy-mm"
OI_spend_city$date_m <- format(OI_spend_city$date, "%Y-%m")

# Drop the observations that are earlier than January 2019 or later than June 2022
start_date <- as.Date("2019-01-01")
end_date <- as.Date("2022-06-30")
OI_spend_city <- OI_spend_city %>%
  filter(date >= start_date & date <= end_date)

# We can now look at the tail of the data to confirm the changes
tail(OI_spend_city)
```
~~~

	year	month	day
	<int>	<int>	<int>
45385	2022	6	26
45386	2022	6	26
45387	2022	6	26
45388	2022	6	26
45389	2022	6	26
45390	2022	6	26

3. Merge

~~~
```{r}
# Load the mapping dataset
OI_city_fips <- read.delim2("OI_city_fips.txt", header = TRUE, sep =",", dec = ".")
OI_city_fips
# Merge the datasets by "cityid"
OI_spend_city_merged <- merge(OI_spend_city, OI_city_fips, by = "cityid")

# Check the first few rows of the merged dataset
head(OI_spend_city_merged)
```
~~~

Description: df [6 × 33]

	cityid	year	month	day	freq	spend_all	spend_aap	spend_acf	spend_aer
	<int>	<int>	<int>	<int>	<chr>	<dbl>	<chr>	<chr>	<chr>
1	1	2020	1	1	d	NA	.	.	.
2	1	2021	6	20	d	-0.0389	.0437	-.0782	-.0227
3	1	2020	7	12	d	-0.1840	-.177	-.453	-.621
4	1	2021	12	30	d	-0.0501	.144	-.196	-.426
5	1	2021	1	21	d	-0.0941	-.0766	-.409	-.613
6	1	2020	5	23	d	-0.2780	-.497	-.609	-.733

Due to the complexity of converting types for columns and merging, we use Python for further cleaning and analysis, so we write out this dataframe as a csv file:

~~~
```{r}
write.csv(OI_spend_city_merged, "OI_spend_city_merged.csv", row.names = FALSE)
```
~~~

4. Generate the median of spend_all growth rate at the year-month-state-city level

It might be better to start doing everything using Python form here

```python
# We first read OI_spend_city_merged.csv generated in R Studio
import pandas as pd
# read OI_spend_city_merged.csv
OI_spend_city_merged = pd.read_csv('OI_spend_city_merged.csv')
OI_spend_city_merged.head()

# convert the data types of all columns that start with 'spend' to float
spend_columns = OI_spend_city_merged.filter(like='spend').columns
display(spend_columns)
```

```python
# we then convert the data types of all columns that start with 'spend' to float
OI_spend_city_merged[spend_columns] = OI_spend_city_merged[spend_columns].apply(pd.to_numeric, errors='coerce')
display(OI_spend_city_merged.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45390 entries, 0 to 45389
Data columns (total 33 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   cityid                45390 non-null  int64
 1   year                  45390 non-null  int64
 2   month                 45390 non-null  int64
 3   day                   45390 non-null  int64
 4   freq                  45390 non-null  object
 5   spend_all             43022 non-null  float64
 6   spend_aap             43022 non-null  float64
 7   spend_acf             43022 non-null  float64
 8   spend_aer             43022 non-null  float64
 9   spend_apg             43022 non-null  float64
 10  spend_durables        43022 non-null  float64
 11  spend_nondurables     43022 non-null  float64
 12  spend_grf             43022 non-null  float64
 13  spend_gen             43022 non-null  float64
 14  spend_hic             43022 non-null  float64
 15  spend_hcs             43022 non-null  float64
 16  spend_inperson        43022 non-null  float64
 17  spend_inpersonmisc    43022 non-null  float64
 18  spend_remoteservices  43022 non-null  float64
 19  spend_sgh             43022 non-null  float64
...
 31  lon                   45390 non-null  float64
 32  city_pop2019          45390 non-null  int64
dtypes: float64(20), int64(7), object(6)
```

```python
# Let's perform the calculation of median spend growth rates for all the 'spend...' columns in Python

# Find all columns that start with 'spend...'
spend_columns = [col for col in OI_spend_city_merged.columns if col.startswith('spend_')]
print(spend_columns)

# Calculate the median of each spend growth rate at the year-month-state-city level
median_spend_growth = OI_spend_city_merged.groupby(['date_m', 'stateabbrev', 'cityname'])[spend_columns].median().reset_index()
median_spend_growth['cityname'] = median_spend_growth['cityname'].str.upper()

# Display the resulting dataframe
median_spend_growth.head()
```

```
['spend_all', 'spend_aap', 'spend_acf', 'spend_aer', 'spend_apg', 'spend_durables', 'spend_nondurables', 'spend_grf', 'spend_gen', 'spend_hic', 'spend_hcs', ':
```

	date_m	stateabbrev	cityname	spend_all	spend_aap	spend_acf	spend_aer	spend_apg	spend_durables	spend_nondurables	...	spend_gen	spend_hic	spe
0	2020-01	AZ	PHOENIX	-0.002080	-0.01590	0.011300	0.04150	-0.0122	-0.02240	-0.01500	...	-0.0115	-0.06020	-0
1	2020-01	AZ	TUCSON	-0.002040	-0.02460	-0.000563	0.06940	0.0019	-0.01620	-0.00547	...	0.0141	-0.06150	0
2	2020-01	CA	BAKERSFIELD	0.005130	0.00810	-0.023800	0.08530	-0.0169	-0.00858	-0.00796	...	-0.0178	-0.00858	-(
3	2020-01	CA	FRESNO	0.000487	0.00251	-0.009880	0.00615	-0.0104	0.00426	-0.00747	...	-0.0213	0.02120	-0
4	2020-01	CA	LOS ANGELES	0.000012	-0.02110	0.005520	-0.02540	-0.0219	-0.01150	-0.01450	...	-0.0191	-0.02960	-0

5 rows × 21 columns

5. Merge Q4 with the csv file
   Due to the complexity of converting date_m into date format and merging, we used Python to do so:

   For this step, we will merge the median_spend_growth and Safegraph_pct_ios dataframes on the 'date_m', 'stateabbrev', and 'cityname' columns after matching the column names.

```python
Safegraph_pct_ios = pd.read_csv('Safegraph_pct_ios.csv')

# Define a function to convert date_m from '2020m1' format to '2020-01'
def convert_date_m(date_m_str):
  # Split the string on 'm' and add '0' padding to single-digit months
  year, month = date_m_str.split('m')
  month = month.zfill(2)  # Ensure month is in '01' format
  return f'{year}-{month}'  # Return the formatted date string

# Apply the conversion function to the date_m column and directly update it
Safegraph_pct_ios['date_m'] = Safegraph_pct_ios['date_m'].apply(convert_date_m)

# We then change the column names to match the OI_spend_city_merged dataframe
Safegraph_pct_ios.rename(columns={'city':'cityname', 'state_abbrev':'stateabbrev'}, inplace=True)
```

```
# Display the head of the dataframe to confirm the changes
Safegraph_pct_ios.head()
```

	date_m	stateabbrev	cityname	android_visits	ios_visits	pct_ios
0	2020-01	AZ	PHOENIX	405512	197490	0.327511
1	2020-02	AZ	PHOENIX	320698	176810	0.355391
2	2020-03	AZ	PHOENIX	298406	163813	0.354406
3	2020-04	AZ	PHOENIX	213844	109350	0.338342
4	2020-05	AZ	PHOENIX	250898	141089	0.359933

Note that right here, we have used Python to clean and merge the datasets, and we'll continue our analysis using R:

Then we

```
# Merge the median_spend_growth and Safegraph_pct_ios dataframes on the date_m, stateabbrev, and cityname columns
merged_Q5 = pd.merge(median_spend_growth, Safegraph_pct_ios, on=['date_m', 'stateabbrev', 'cityname'])
display(merged_Q5)
```

	date_m	stateabbrev	cityname	spend_all	spend_aap	spend_acf	spend_aer	spend_apg	spend_durables	spend_nondurables	...	spend_inperson	spend_inp
0	2020-01	AZ	PHOENIX	-0.002080	-0.01590	0.011300	0.04150	-0.0122	-0.02240	-0.01500	...	-0.004870	
1	2020-01	AZ	TUCSON	-0.002040	-0.02460	-0.000563	0.06940	0.0019	-0.01620	-0.00547	...	-0.006750	
2	2020-01	CA	BAKERSFIELD	0.005130	0.00810	-0.023800	0.08530	-0.0169	-0.00858	-0.00796	...	0.002930	
3	2020-01	CA	FRESNO	0.000487	0.00251	-0.009880	0.00615	-0.0104	0.00426	-0.00747	...	-0.000539	
4	2020-01	CA	LOS ANGELES	0.000012	-0.02110	0.005520	-0.02540	-0.0219	-0.01150	-0.01450	...	-0.009690	

5 rows × 24 columns

```
# Report the summary statistics of the columns spend_all and spend_inperson
summary_statistics = merged_Q5[['spend_all', 'spend_inperson']].describe()
summary_statistics
```

	spend_all	spend_inperson
count	1440.000000	1440.000000
mean	0.008324	-0.151338
std	0.123813	0.200066
min	-0.449500	-0.752500
25%	-0.056125	-0.281000
50%	0.014075	-0.125750
75%	0.087387	0.002990
max	0.346000	0.332000

6. "post_ATT"

```
# Convert date_m to datetime to make comparison possible
merged_data_Python['date_m'] = pd.to_datetime(merged_data_Python['date_m'])

# Generate the post_ATT dummy variable
# For dates in May 2021 or later, the dummy variable should be equal to 1
merged_data_Python['post_ATT'] = (merged_data_Python['date_m'] >= '2021-05-01').astype(int)

display(merged_data_Python.head())
display(merged_data_Python.tail())
```

	date_m	stateabbrev	cityname	median_spend_all	median_spend_inperson	android_visits	ios_visits	pct_ios	post_ATT
0	2020-01-01	AZ	PHOENIX	-0.002080	-0.004870	405512	197490	0.327511	0
1	2020-01-01	AZ	TUCSON	-0.002040	-0.006750	222473	87921	0.283256	0
2	2020-01-01	CA	BAKERSFIELD	0.005130	0.002930	111561	74278	0.399690	0
3	2020-01-01	CA	FRESNO	0.000487	-0.000539	116912	78379	0.401345	0
4	2020-01-01	CA	LOS ANGELES	0.000012	-0.009690	362221	215135	0.372621	0

	date_m	stateabbrev	cityname	median_spend_all	median_spend_inperson	android_visits	ios_visits	pct_ios	post_ATT
1495	2022-06-01	TX	SAN ANTONIO	0.12300	0.12600	362951	310776	0.461279	1
1496	2022-06-01	UT	SALT LAKE CITY	0.14350	0.02785	34561	23830	0.408111	1
1497	2022-06-01	VA	VIRGINIA BEACH	0.19100	0.16500	62298	53484	0.461937	1
1498	2022-06-01	WA	SEATTLE	0.08865	0.02175	48083	28326	0.370715	1
1499	2022-06-01	WI	MILWAUKEE	0.21650	0.01745	49539	48463	0.494510	1

7. Median of pct_ios per stateabbrev & cityname

```
# filter the date, use groupby to calculate the median of pct_ios, and rename the column
pre_december_2020 = merged_Q6[merged_Q6['date_m'] < '2020-12-01']
treatment_intensity = pre_december_2020.groupby(['stateabbrev', 'cityname'])['pct_ios'].median().reset_index()
treatment_intensity.rename(columns={'pct_ios': 'treatment_intensity'}, inplace=True)
display(treatment_intensity.head())

# Then we merge this weith the merged_Q6 dataframe
merged_Q7 = pd.merge(merged_Q6, treatment_intensity, on=['stateabbrev', 'cityname'])
display(merged_Q7.head()) # If my understanding is correct, this would result in the same treatment_intensity for all rows of the same stateabbrev & cityname
```

	stateabbrev	cityname	treatment_intensity
0	AZ	PHOENIX	0.368868
1	AZ	TUCSON	0.335710
2	CA	BAKERSFIELD	0.436458
3	CA	FRESNO	0.435484
4	CA	LOS ANGELES	0.395875

s	...	spend_remoteservices	spend_sgh	spend_tws	spend_retail_w_grocery	spend_retail_no_grocery	android_visits	ios_visits	pct_ios	post_ATT	treatment_intensity
0	...	0.00941	-0.0470	0.0444	-0.0145	-0.01750	405512	197490	0.327511	0	0.368868
7	...	-0.01280	-0.0651	-0.0284	-0.0112	-0.00813	222473	87921	0.283256	0	0.335710
6	...	-0.00584	-0.1130	0.0679	-0.0177	-0.03600	111561	74278	0.399690	0	0.436458
7	...	-0.01420	0.0839	-0.0719	-0.0048	0.00341	116912	78379	0.401345	0	0.435484
0	...	0.01990	-0.0436	0.0108	-0.0138	-0.02280	362221	215135	0.372621	0	0.395875

8. Regression

```
merged_Q7['date_m'] = pd.to_datetime(merged_Q7['date_m']).dt.to_period('M')
display(merged_Q7.head())

# Create the interaction term variable
merged_Q7['interaction'] = merged_Q7['post_ATT'] * merged_Q7['treatment_intensity']
display(merged_Q7.head())

%pip install patsy
from patsy import dmatrices
import statsmodels.api as sm
# Convert 'year_month' from Period to string for compatibility with patsy
merged_Q7['year_month_str'] = merged_Q7['date_m'].astype(str)

# Update the formula to use the string version of 'year_month'
formula = 'spend_nondurables ~ 1 + interaction + C(year_month_str) + C(cityname)'

# Generate the design matrices
y, X = dmatrices(formula, merged_Q7, return_type='dataframe')

# Fit the model using Ordinary Least Squares
model = sm.OLS(y, X).fit()

# Show the summary of the regression model
model.summary().tables[0]
```

	OLS Regression Results		
Dep. Variable:	spend_nondurables	R-squared:	0.865
Model:	OLS	Adj. R-squared:	0.857
Method:	Least Squares	F-statistic:	112.9
Date:	Sat, 06 Apr 2024	Prob (F-statistic):	0.00
Time:	15:17:20	Log-Likelihood:	2512.0
No. Observations:	1440	AIC:	-4868.
Df Residuals:	1362	BIC:	-4457.
Df Model:	77		
Covariance Type:	nonrobust		

9. Regression

```
# Create state and year-month interaction fixed effects
merged_Q7['state_year_month'] = merged_Q7['stateabbrev'] + '_' + merged_Q7['year_month_str']

# Update the formula to include interaction of state and year-month, and control for city fixed effects
formula_q9 = 'spend_nondurables ~ 1 + interaction + C(cityname) + C(state_year_month)'

# Generate the design matrices for question 9
y_q9, X_q9 = dmatrices(formula_q9, merged_Q7, return_type='dataframe')

# Fit the model using Ordinary Least Squares
model_q9 = sm.OLS(y_q9, X_q9).fit()

# Show the summary of the regression model for question 9
model_q9.summary().tables[0]
```

	OLS Regression Results		
Dep. Variable:	spend_nondurables	R-squared:	0.952
Model:	OLS	Adj. R-squared:	0.874
Method:	Least Squares	F-statistic:	12.18
Date:	Sat, 06 Apr 2024	Prob (F-statistic):	6.24e-166
Time:	15:17:50	Log-Likelihood:	3254.1
No. Observations:	1440	AIC:	-4728.
Df Residuals:	550	BIC:	-35.67
Df Model:	889		
Covariance Type:	nonrobust		

10. Online spending

```
# Generate a new variable 'online_spending'
merged_Q7['online_spending'] = merged_Q7['spend_all'] - merged_Q7['spend_inperson']

# Display the first few rows to confirm the creation of the new variable
merged_Q7[['spend_all', 'spend_inperson', 'online_spending']].head()
```

	spend_all	spend_inperson	online_spending
0	-0.002080	-0.004870	0.002790
1	-0.002040	-0.006750	0.004710
2	0.005130	0.002930	0.002200
3	0.000487	-0.000539	0.001026
4	0.000012	-0.009690	0.009702

11. Regression

```
# Update the formula to use 'online_spending' as the outcome variable
formula_q11 = 'online_spending ~ 1 + interaction + C(cityname) + C(state_year_month)'

# Generate the design matrices for question 11
y_q11, X_q11 = dmatrices(formula_q11, merged_Q7, return_type='dataframe')
```

```python
# Fit the model using Ordinary Least Squares
model_q11 = sm.OLS(y_q11, X_q11).fit()

# Show the summary of the regression model for question 11
model_q11.summary().tables[0]
```

OLS Regression Results

Dep. Variable:	online_spending	R-squared:	0.969
Model:	OLS	Adj. R-squared:	0.919
Method:	Least Squares	F-statistic:	19.27
Date:	Sat, 06 Apr 2024	Prob (F-statistic):	4.06e-215
Time:	15:22:36	Log-Likelihood:	3628.5
No. Observations:	1440	AIC:	-5477.
Df Residuals:	550	BIC:	-784.5
Df Model:	889		
Covariance Type:	nonrobust		

12. Regression on all 19

```python
# To run the regression for all 19 spending variables, we first need to identify all those variables
# We will assume they follow a naming convention starting with "spend_" as seen before

# List all columns starting with "spend_" but not including 'spend_all' as it is a total spending variable
spending_vars = [col for col in merged_Q7.columns if col.startswith('spend_')]

# Define the base formula for the regression
base_formula = ' ~ 1 + interaction + C(cityname) + C(state_year_month)'

# We'll create a table to store the coefficients, p-values, and t-stats for post_ATT x treatment_intensity for each spending variable
coefficients_table = []

# Run the regression for each spending variable and collect the required statistics
for var in spending_vars:
  formula = f'{var}' + base_formula
  y, X = dmatrices(formula, merged_Q7, return_type='dataframe')
  model = sm.OLS(y, X).fit()
  coefficient = model.params['interaction']
  p_value = model.pvalues['interaction']
  t_stat = model.tvalues['interaction']
  coefficients_table.append((var, coefficient, p_value, t_stat))

# Convert the list to a DataFrame
coefficients_df = pd.DataFrame(coefficients_table, columns=['Variable', 'Coefficient', 'P-value', 'T-statistic'])

coefficients_df
```

	Variable	Coefficient	P-value	T-statistic
0	spend_all	-0.185966	1.224523e-02	-2.513307
1	spend_aap	-0.174117	3.608377e-01	-0.914533
2	spend_acf	-0.519788	5.197285e-07	-5.079252
3	spend_aer	-0.293780	1.911925e-01	-1.308671
4	spend_apg	-0.220833	1.049593e-01	-1.623951
5	spend_durables	0.005477	9.588315e-01	0.051644
6	spend_nondurables	-0.274843	1.869839e-03	-3.125319
7	spend_grf	-0.122577	1.366014e-01	-1.490746
8	spend_gen	-0.353973	8.380201e-03	-2.645915
9	spend_hic	0.293519	1.388786e-01	1.482129
10	spend_hcs	-0.434883	7.421042e-04	-3.392614
11	spend_inperson	-0.443935	5.878219e-07	-5.054710
12	spend_inpersonmisc	-0.285977	4.808945e-02	-1.980995
13	spend_remoteservices	-0.299461	1.399965e-03	-3.211006
14	spend_sgh	0.490974	8.135207e-02	1.746099
15	spend_tws	-0.309678	2.018031e-02	-2.329751
16	spend_retail_w_grocery	-0.100302	3.148501e-01	-1.006016
17	spend_retail_no_grocery	-0.007796	9.539220e-01	-0.057809

13. ATT (Apple's App Tracking Transparency) can be considered a good 'natural experiment' because it was an externally imposed policy change that affected user privacy and data sharing practices across all iOS devices. Since it was implemented at a specific point in time to all users and was not influenced by the users themselves or the app developers, it created a clear before-and-after scenario. This can be

exploited to measure causal effects as it mimics a randomized controlled trial where the 'treatment' (the implementation of ATT) was not correlated with other factors that might affect spending, making it easier to isolate the effect of ATT on spending behavior.

14. In the given tasks, we are interested in the interaction between post-ATT and treatment intensity, which inherently includes both post-ATT and treatment intensity within it. Since we are looking at the interaction effect specifically, there is no need to control for the individual effects of post-ATT and treatment intensity; the interaction term itself represents the combined effect of these two variables when they occur together. Controlling them separately would be redundant and could potentially distort the estimation of the interaction effect.
15. Two alternative rationales for the model specification that includes the interaction term post_ATT × treatment_intensity could be:
    a. Positive coefficient: The rationale for a positive coefficient could be that the implementation of ATT has led to more targeted and efficient use of advertising budgets. Advertisers may be spending more efficiently and getting better returns on investment, which could result in increased spending due to better ad performance and conversion rates.
    b. Negative coefficient: Conversely, a negative coefficient might be explained by the reduction in data available for targeting ads, which could decrease their effectiveness and thereby reduce spending. Advertisers might be less willing to spend on ads that are less targeted and therefore less likely to convert, leading to an overall reduction in ad spending.

    For both rationales, one would need to cite credible sources that provide evidence or arguments supporting these explanations. For example, industry reports, academic studies, or data released by advertising platforms discussing the effects of data privacy regulations on advertising effectiveness and spending could be relevant.
16. The interpretation of the coefficient on post_ATT × treatment_intensity should include both the statistical significance, which speaks to whether the observed effect is likely to be a real one rather than a result of random chance, and the economic significance, which considers the size of the effect and its practical implications for stakeholders.
    a. Statistical Significance: If the p-value is less than the conventional threshold (usually 0.05), the coefficient is statistically significant, which means there is a low probability that the observed relationship is due to random chance.
    b. Economic Significance: This relates to the magnitude of the coefficient and whether it represents a big enough change in spending to be considered important by industry standards. For example, even a statistically significant result may have a very small coefficient, which might not be economically meaningful if the actual change in spending is minuscule.

    For task 11, the coefficient on the interaction term was found to be statistically significant with a large enough effect size, it would suggest that the ATT policy change meaningfully affected online spending behavior when interacting with the intensity of treatment (iOS user share in this case).

Part 2

1. Several factors contributed to the rapid development of Fintech in China:
    - **Technological Advances**: With widespread digital adoption and over 30% of the nation's population using Internet payment systems, technological advances provided a strong foundation for Fintech growth.
    - **E-Commerce Development**: A highly developed e-commerce sector led to a natural evolution and integration of financial services online.
    - **Latent Demand for Inclusive Finance**: There was a significant latent demand for inclusive financial services, as traditional banks often underserved small businesses, microenterprises, and rural populations.
    - **Regulatory Environment**: The government's efforts to establish a regulatory framework for Fintech balanced innovation and societal stability, even though it was still developing and some areas remained under-regulated.
    - **Market Voids**: Institutional voids in the traditional banking sector, like the lack of an efficient credit profiling mechanism and difficulties SMEs faced in securing funding, created opportunities for Fintech firms to fill these gaps.
2. Ant Financial's main advantages were:
    - **Broad Range of Services**: They created a financial ecosystem offering services such as payments, wealth management, banking, credit scoring, and insurance.
    - **Technology and Data Analytics**: Ant Financial utilized advanced data analytics and artificial intelligence for credit profiling and risk management, which enabled them to efficiently serve a large user base with customized financial products.
    - **Massive User Base**: With 451 million active users just in the payment sector, Ant Financial had a vast amount of user data to refine and develop their services.
    - **Innovative Credit Profiling**: Zhima Credit developed an innovative credit scoring system that used diverse data sources, facilitating the provision of inclusive finance.
    - **Operational Efficiency**: Their robust cloud technology platform and big-data-based fraud risk management capabilities enabled them to process transactions at scale and maintain high security with low operational costs.

- **Rural and International Strategies**: Ant Financial had targeted strategies to extend their services to rural areas in China and were actively pursuing international expansion through investments and strategic partnerships.
- **Scenario-Based Prototyping**: They effectively used scenario-based strategies to test and refine fintech solutions, which attracted users and helped mainstream their technology.