In [1]:
```
# import statements
import pandas as pd
from textblob import TextBlob
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
import string
import re
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
import networkx as nx
from matplotlib.lines import Line2D
import random
import statsmodels.api as sm
from stargazer.stargazer import Stargazer
from IPython.core.display import HTML
import geopandas as gpd
```

Introduction

Public opinion evolves over time and often demographic factors cannot capture the variation between surveys. We used 4-years of Taiwanese public opinion data, coupled with data scraped from the leading Taiwanese newspaper the *Taipei Times* to help explain the variation in the public's perceptions of Mainland China. The questions in the surveys that ask about respondent's political affiliation, ethnic identity, and other demographic factors are insufficient to deal with non-stationary trends in the data. For example, researchers found that the 2019 Hong Kong protests substantially soured Taiwanese perceptions of Mainland China (Rich et al., 2020).

Given the continued high tensions between Taiwan and Mainlnad China throughout the US-China Trade War (Everington, 2019), Hong Kong protests (Kwan, 2022), and COVID-19 pandemic (Blanchard, 2022), as well as the military and diplomatic flyovers over the Taiwanese Strait (Tan & Molloy, 2022), little statistical research has been conducted into how recent events have shaped the growing Taiwanese identity and strategic industry economic decoupling of Taiwan and the United States from China in recent years (Cheng et al., 2022). With a focus on specifically the Hong Kong protests in early 2019, we aim to answer the following question: *Do external events impact Taiwanese public perceptions of Mainland China?*.

We answer this question in three parts:

1. We use four-years of Taiwanese public opinion data and visualize changes in public opinion surrounding significant events.
2. We scraped *Taipei Times* data from the two week period preceding each survey and account for significant events via sentiment and content analysis and topic modeling.
3. We combine the public opinion data and scraped data to analyze the impact of external events.

Overall, we find that significant variation occurs between surveys. The scraped *Taipei Times* data seems to help us find that salient issues like the Hong Kong protests, economic and infrastructure concerns, and Covid-19 sway public opinion. Additionally, general sentiment and classification of the articles are important variables in our analysis. Researchers in the future must take into account topical events when analysis time-series or panel data, which is often difficult using just the survey data itself.

We rely on four dependent variables and use the survey data and scraped news data to find several independent variables. Below, we include a table identifying our dependent and independent variables.

We select four dependent variables, all of which proxy Taiwanese perceptions of Mainland China. The first variable approval of Tsai's cross-straight relations policy identifies how Taiwanese feel about their president's handeling of Taiwanese-Chinese relations. The second variable is the number of respondents who identified Taiwanese-Chinese relations (often referred to as Cross-Strait Relations) as the most important issue. The third variable identifies whether respondents view themselves as Taiwanese, Chinese, or both. The final variable is related to the political status of Taiwan and whether respondents want independence, the status quo, or unification with Mainland China.

Dependent Variables

Variable No.	Variable Name	Variables ID
1	Approval of President's (Tsai) Cross-Straight Relations Policy	tsai_csr
2	Most Import Issue (Cross-Straight Relations)	most_import
3	Identity: Chinese vs Taiwanese	identity
4	Taiwanese Reunification with China	status

Our independent variables are a series of demographic factors found in the surveys (variables 1-9) and external variables. The remaining variables (10-13) include the number of articles in the two-week period preceding each survey, the average sentiment per survey, the number of articles classified into different categories (i.e. news, editorials, sports), and the number of articles per topic (identified by topic analysis in the previous section).

Independent Variables

Variable No.	Variable Name	Variables ID
1	gender identification	gender
2	political party affiliation	party
3	presidential performance (Tsai)	tsai_perf
4	perceptions of recent past economic performance	past_econ
5	perceptions of future economic performance	fut_econ
6	education level	education
7	Region respondent lives in	location
8	2016 or 2020 presidential vote	vote
9	survey respondent took	survey
10	article count	article_count
11	text sentiment	text_sentiment
12	article classification	misc.
13	article topic	misc.

Part 1: TEDS Survey Data

This section analyzes the Taiwan's Election and Democratization Study (TEDS) survey data, from early 2017 to the end of 2020. TEDS is a public opinion lab at the Department of Humanities and Social Science of the Taiwanese government's National Science and Technology Council. Most of the public opinion data conducted in Taiwan, even surveys developed by external groups, are conducted by TEDS (TEDS, no date). Typically, TEDS conducts 3-6 public opinion surveys each year, asking about relations with China, economic issues, and salient political issues. To access the data, we made an account with TEDS and request each survey from 2017-2020 for download. The data was cleaned in the **TEDS Data Wrangling Notebook** since the cleaning was quite extensive.

The table below shows the dates for each survey. The first two surveys are named 1.1 and 1.2 because they were two seperate surveys conducted during the same period.

Survey Dates

Survey No.	Survey Date
1.1	March 23, 2017
1.2	March 23, 2017
2	May 19, 2017
3	June 9, 2017
4	Sept. 21, 2017
5	Dec 14, 2017
6	Jan 2, 2018
7	March 21, 2018
8	May 10, 2018
9	June 7, 2018
10	Sept. 25, 2018
11	Dec 18, 2018
12	May 9, 2019
13	June 11, 2019
14	Sept. 24, 2019
15	Dec 11, 2019
16	March 26, 2020
17	June 2, 2020
18	Sept. 22, 2020
19	Dec 22, 2020

In this section, we clean and prepare the data so we can aggregate each dependent variable into line charts and we create a geospatial dataframe so that we can visualize the data in maps. We specifically choose to visualize the change in our four dependent variables before and after the Hong Kong protests to demonstrate that significant political events alter Taiwanese public perceptions of Mainland China.

Cleaned Survey Disaggregated Data

Clean and Prepare Disaggregated Data

We begin by reading in the already cleaned TEDS survey data and keeping a copy without modifications.

In [2]:
```
# read in new data
df_survey_clean = pd.read_csv('Data/df_survey_clean.csv')

# keep copy of old data
df_survey = df_survey_clean.copy()
```

We need to clean and prepare the data so we can aggregate it into percentages per survey. We add identifying language to each of our dependent variables, otherwise when we create dummy variables it is difficult to differentiate between variables. We drop na values for each dependent variable, and we create dummy variables for the range of responses to each dependent variable.

In [3]:
```
# add identifying language

# tsai csr
df_survey_clean['tsai_csr'] = 'tsai_csr ' + df_survey_clean['tsai_csr'].astype(str)

# csr most important
df_survey_clean['most_import'] = 'most_import ' + df_survey_clean['most_import'].astype(str)

# identity
df_survey_clean['identity'] = 'identity ' + df_survey_clean['identity'].astype(str)

# reunifcation
df_survey_clean['status'] = 'status ' + df_survey_clean['status'].astype(str)
```

In [4]:
```
# drop na values

# drop tsai_csr na rows
df_tsai_csr = df_survey_clean.dropna(subset=['tsai_csr'])

# drop most important na rows
df_most_impot = df_survey_clean.dropna(subset=['most_import'])

# drop status na rows
df_status = df_survey_clean.dropna(subset=['status'])

# drop identity na rows
df_identity = df_survey_clean.dropna(subset=['identity'])
```

In [5]:
```
# tsai csr aggregate the data

# tsai csr
dummies_tsaicsr = pd.get_dummies(df_tsai_csr['tsai_csr'])

# basic df
df_tsai_csr = df_tsai_csr[['survey']].copy()

# join dummies
df_tsai_csr = df_tsai_csr.join(dummies_tsaicsr)
```

In [6]:
```
# csr most important aggregate the data

# tsai csr
dummies_mi = pd.get_dummies(df_most_impot['most_import'])

# basic df
df_most_impot = df_most_impot[['survey']].copy()

# join dummies
df_most_impot = df_most_impot.join(dummies_mi)
```

In [7]:
```
# identity aggregate the data

# identity
dummies_i = pd.get_dummies(df_identity['identity'])

# basic df
df_identity = df_identity[['survey']].copy()

# join dummies
df_identity = df_identity.join(dummies_i)
```

In [8]:
```
# status aggregate data

# status
dummies_s = pd.get_dummies(df_status['status'])

# basic df
df_status = df_status[['survey']].copy()

# join dummies
```

```
df_status = df_status.join(dummies_s)
```

For each dependent variable dataset, we calculate the percentages of each dependent and in-survey independent variable so that we can analyze aggregated measurements.

In [9]:
```
# Tsai CSR dataset

# aggregate dataset
df_count = df_tsai_csr.groupby(['survey'])[['survey']].count()
df_tsai_csr = df_tsai_csr.groupby(['survey']).sum()
df_tsai_csr = pd.concat([df_count, df_tsai_csr], axis=1)

# transpose dataframe
tsai_csr_flat = df_tsai_csr.T

# calculate percentages
tsai_csr_flat[1.1] = (tsai_csr_flat[1.1] / 1101) * 100
tsai_csr_flat[1.2] = (tsai_csr_flat[1.2] / 1218) * 100
tsai_csr_flat[2.0] = (tsai_csr_flat[2.0] / 1203) * 100
tsai_csr_flat[3.0] = (tsai_csr_flat[3.0] / 1230) * 100
tsai_csr_flat[4.0] = (tsai_csr_flat[4.0] / 1231) * 100
tsai_csr_flat[5.0] = (tsai_csr_flat[5.0] / 1268) * 100
tsai_csr_flat[6.0] = (tsai_csr_flat[6.0] / 1008) * 100
tsai_csr_flat[7.0] = (tsai_csr_flat[7.0] / 1241) * 100
tsai_csr_flat[8.0] = (tsai_csr_flat[8.0] / 1107) * 100
tsai_csr_flat[9.0] = (tsai_csr_flat[9.0] / 1214) * 100
tsai_csr_flat[10.0] = (tsai_csr_flat[10.0] / 1200) * 100
tsai_csr_flat[11.0] = (tsai_csr_flat[11.0] / 1219) * 100
tsai_csr_flat[12.0] = (tsai_csr_flat[12.0] / 1100) * 100
tsai_csr_flat[13.0] = (tsai_csr_flat[13.0] / 1365) * 100
tsai_csr_flat[14.0] = (tsai_csr_flat[14.0] / 1240) * 100
tsai_csr_flat[15.0] = (tsai_csr_flat[15.0] / 1260) * 100
tsai_csr_flat[16.0] = (tsai_csr_flat[16.0] / 1299) * 100
tsai_csr_flat[17.0] = (tsai_csr_flat[17.0] / 1217) * 100
tsai_csr_flat[18.0] = (tsai_csr_flat[18.0] / 1214) * 100
tsai_csr_flat[19.0] = (tsai_csr_flat[19.0] / 1254) * 100

# delete survey 2, 6, 8, 12
tsai_csr_flat = tsai_csr_flat.drop([2.0, 6.0, 8.0, 12.0], axis=1)

# re-transpose dataframe
tsai_csr_df = tsai_csr_flat.T
```

In [10]:
```
# CSR key issue dataset

# aggregate dataset
df_count = df_most_impot.groupby(['survey'])[['survey']].count()
df_most_impot = df_most_impot.groupby(['survey']).sum()
df_most_impot = pd.concat([df_count, df_most_impot], axis=1)

# transpose dataframe
most_impot_flat = df_most_impot.T

# calculate percentages
most_impot_flat[1.1] = (most_impot_flat[1.1] / 1101) * 100
most_impot_flat[1.2] = (most_impot_flat[1.2] / 1218) * 100
most_impot_flat[2.0] = (most_impot_flat[2.0] / 1203) * 100
most_impot_flat[3.0] = (most_impot_flat[3.0] / 1230) * 100
most_impot_flat[4.0] = (most_impot_flat[4.0] / 1231) * 100
most_impot_flat[5.0] = (most_impot_flat[5.0] / 1268) * 100
most_impot_flat[6.0] = (most_impot_flat[6.0] / 1008) * 100
most_impot_flat[7.0] = (most_impot_flat[7.0] / 1241) * 100
most_impot_flat[8.0] = (most_impot_flat[8.0] / 1107) * 100
most_impot_flat[9.0] = (most_impot_flat[9.0] / 1214) * 100
most_impot_flat[10.0] = (most_impot_flat[10.0] / 1200) * 100
most_impot_flat[11.0] = (most_impot_flat[11.0] / 1219) * 100
most_impot_flat[12.0] = (most_impot_flat[12.0] / 1100) * 100
most_impot_flat[13.0] = (most_impot_flat[13.0] / 1365) * 100
most_impot_flat[14.0] = (most_impot_flat[14.0] / 1240) * 100
most_impot_flat[15.0] = (most_impot_flat[15.0] / 1260) * 100
most_impot_flat[16.0] = (most_impot_flat[16.0] / 1299) * 100
most_impot_flat[17.0] = (most_impot_flat[17.0] / 1217) * 100
most_impot_flat[18.0] = (most_impot_flat[18.0] / 1214) * 100
most_impot_flat[19.0] = (most_impot_flat[19.0] / 1254) * 100

# delete survey 2, 6, 12
most_impot_flat = most_impot_flat.drop([2.0, 6.0, 12.0], axis=1)

# re-transpose dataframe
most_impot_df = most_impot_flat.T
```

In [11]:
```
# identity dataset

# aggregate dataset
df_count = df_identity.groupby(['survey'])[['survey']].count()
df_identity = df_identity.groupby(['survey']).sum()
df_identity = pd.concat([df_count, df_identity], axis=1)
```

```
# transpose dataframe
identity_flat = df_identity.T

# calculate percentages
identity_flat[1.1] = (identity_flat[1.1] / 1101) * 100
identity_flat[1.2] = (identity_flat[1.2] / 1218) * 100
identity_flat[2.0] = (identity_flat[2.0] / 1203) * 100
identity_flat[3.0] = (identity_flat[3.0] / 1230) * 100
identity_flat[4.0] = (identity_flat[4.0] / 1231) * 100
identity_flat[5.0] = (identity_flat[5.0] / 1268) * 100
identity_flat[6.0] = (identity_flat[6.0] / 1008) * 100
identity_flat[7.0] = (identity_flat[7.0] / 1241) * 100
identity_flat[8.0] = (identity_flat[8.0] / 1107) * 100
identity_flat[9.0] = (identity_flat[9.0] / 1214) * 100
identity_flat[10.0] = (identity_flat[10.0] / 1200) * 100
identity_flat[11.0] = (identity_flat[11.0] / 1219) * 100
identity_flat[12.0] = (identity_flat[12.0] / 1100) * 100
identity_flat[13.0] = (identity_flat[13.0] / 1365) * 100
identity_flat[14.0] = (identity_flat[14.0] / 1240) * 100
identity_flat[15.0] = (identity_flat[15.0] / 1260) * 100
identity_flat[16.0] = (identity_flat[16.0] / 1299) * 100
identity_flat[17.0] = (identity_flat[17.0] / 1217) * 100
identity_flat[18.0] = (identity_flat[18.0] / 1214) * 100
identity_flat[19.0] = (identity_flat[19.0] / 1254) * 100

# delete survey 12
identity_flat = identity_flat.drop([12.0], axis=1)

# re-transpose dataframe
identity_df = identity_flat.T
```

In [12]:
```
# status dataset

# aggregate dataset
df_count = df_status.groupby(['survey'])[['survey']].count()
df_status_ag = df_status.groupby(['survey']).sum()
df_status = pd.concat([df_count, df_status_ag], axis=1)

# transpose dataframe
status_flat = df_status.T

# calculate percentages
status_flat[1.1] = (status_flat[1.1] / 1101) * 100
status_flat[1.2] = (status_flat[1.2] / 1218) * 100
status_flat[2.0] = (status_flat[2.0] / 1203) * 100
status_flat[3.0] = (status_flat[3.0] / 1230) * 100
status_flat[4.0] = (status_flat[4.0] / 1231) * 100
status_flat[5.0] = (status_flat[5.0] / 1268) * 100
status_flat[6.0] = (status_flat[6.0] / 1008) * 100
status_flat[7.0] = (status_flat[7.0] / 1241) * 100
status_flat[8.0] = (status_flat[8.0] / 1107) * 100
status_flat[9.0] = (status_flat[9.0] / 1214) * 100
status_flat[10.0] = (status_flat[10.0] / 1200) * 100
status_flat[11.0] = (status_flat[11.0] / 1219) * 100
status_flat[12.0] = (status_flat[12.0] / 1100) * 100
status_flat[13.0] = (status_flat[13.0] / 1365) * 100
status_flat[14.0] = (status_flat[14.0] / 1240) * 100
status_flat[15.0] = (status_flat[15.0] / 1260) * 100
status_flat[16.0] = (status_flat[16.0] / 1299) * 100
status_flat[17.0] = (status_flat[17.0] / 1217) * 100
status_flat[18.0] = (status_flat[18.0] / 1214) * 100
status_flat[19.0] = (status_flat[19.0] / 1254) * 100

# delete survey 12
status_flat = status_flat.drop([8.0, 12.0], axis=1)

# re-transpose dataframe
status_df = status_flat.T
```

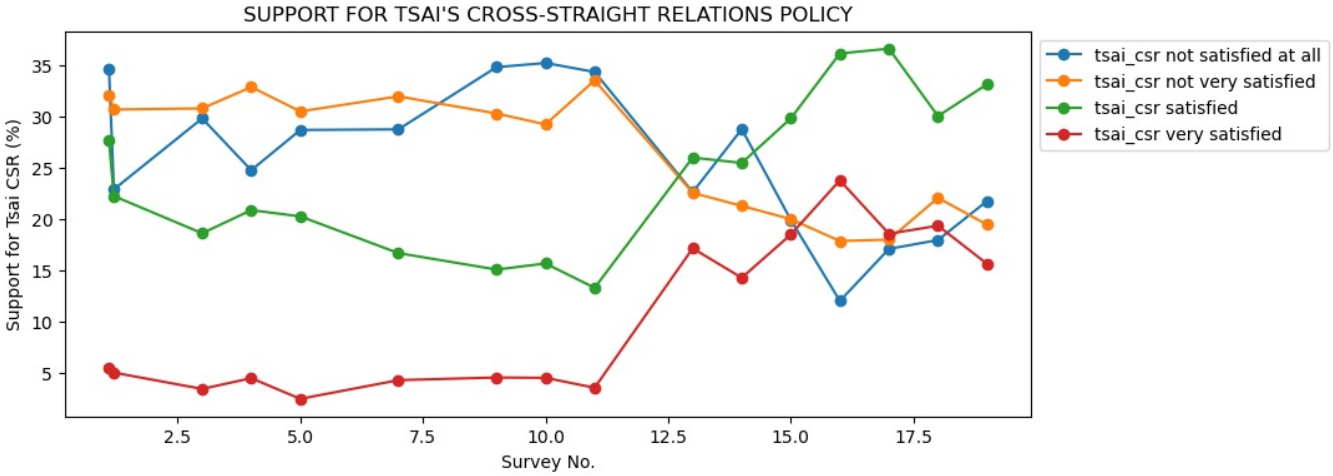
Visualizations: Line Charts

We create four line charts - one for each dependent variable so that we can see variation in responses over time. The results are clear that there is significant external variation in the data.

Tsai's Cross-Strait Relations Policy

In [13]:
```
# Tsai CSR
tsai_csr_line = tsai_csr_df[['tsai_csr not satisfied at all',
                             'tsai_csr not very satisfied',
                             'tsai_csr satisfied',
                             'tsai_csr very satisfied']].copy()

# line chart
tsai_csr_line.plot.line(marker = 'o', figsize=(10,4))
```

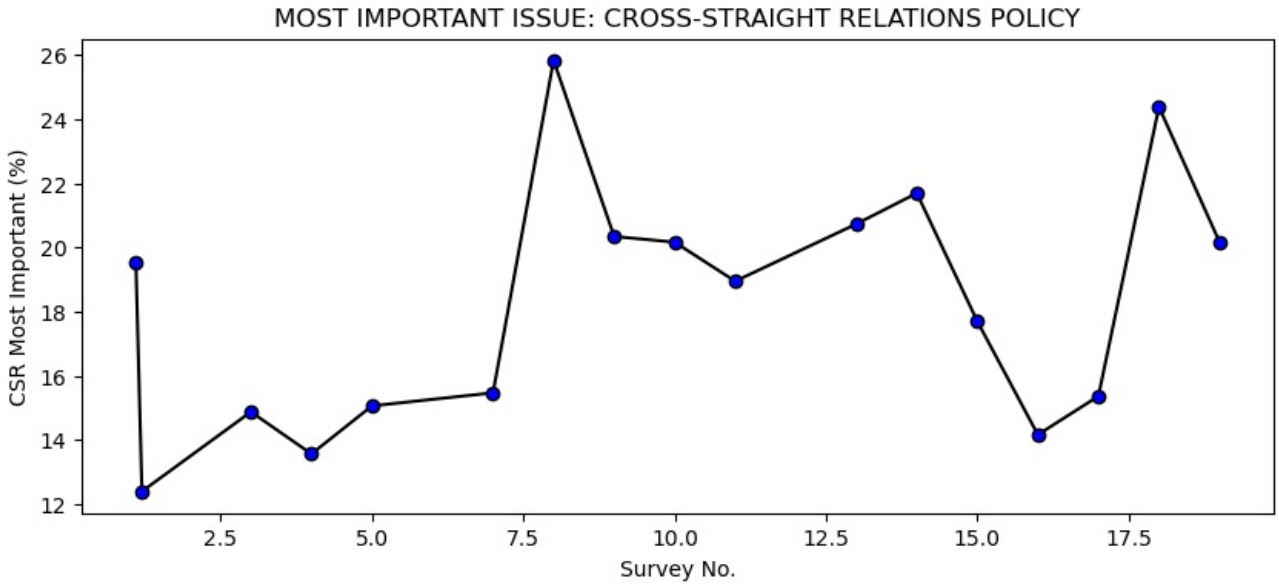
```
plt.xlabel("Survey No.")
plt.ylabel("Support for Tsai CSR (%)")
plt.legend(bbox_to_anchor=(1,1), loc="upper left")
plt.title("SUPPORT FOR TSAI'S CROSS-STRAIGHT RELATIONS POLICY");
```



The figure above shows support for Taiwanese President Tsai's cross-strait relations policy. While a significant portion of respondents were not satisfied or not satisfied at all with her policy in the first half of surveys, this trend shifted around survey twelve where a plurality were satisfied with her policy. Most likely, external events shifted public opinion.

Cross-Strait Relations: Most Important Issue

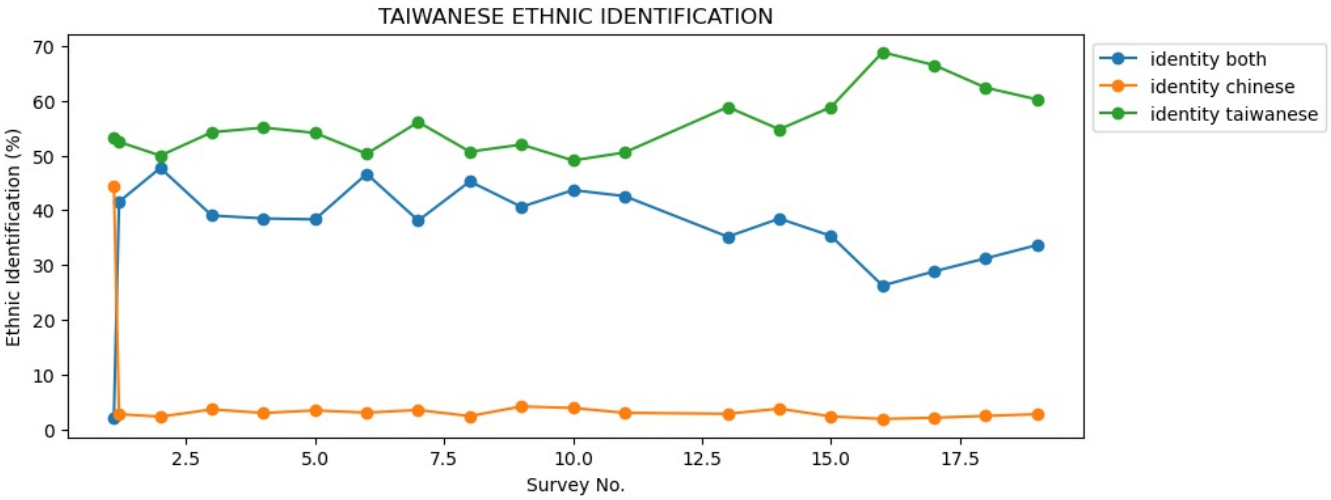
In [14]:
```
# CSR Most Important
csr_line = most_impot_df[['most_import cross-strait relations']].copy()

# line chart
csr_line.plot.line(color = 'black', marker = 'o', markerfacecolor = 'blue', legend=None, figsize=(10,4))
plt.xlabel("Survey No.")
plt.ylabel("CSR Most Important (%)")
plt.title("MOST IMPORTANT ISSUE: CROSS-STRAIGHT RELATIONS POLICY");
```



The line chart above shows whether respondents believed cross-strait relations were the most important issue at that time. While survey 1 begins at almost 20%, there is a downward trend, which spikes upward around survey eight to 25%. The percentage who identified cross-strait relations as most important moderated, then fell to 14%, then shifted upwards. Evidently, there is significant variation between respondent's classification of cross-strait relations as the most important issue.

Taiwanese Identity

In [15]:
```
# Identity
identity_line = identity_df[['identity both', 'identity chinese', 'identity taiwanese']].copy()

# line chart
identity_line.plot.line(marker = 'o', figsize=(10,4))
plt.xlabel("Survey No.")
plt.ylabel("Ethnic Identification (%)")
plt.legend(bbox_to_anchor=(1,1), loc="upper left")
plt.title("TAIWANESE ETHNIC IDENTIFICATION");
```



The line chart above shows the percentage of respondents who identify as Taiwanese, Chinese, or both. In the first survey, about 45% of respondents identify as Chinese, but this falls substantially and remains below 5% for the remainder of our analysis. Eventually, a clear majority of respondents identify as Taiwanese and less classify themselves as both.

Unification with China

In [1]:
```
# Status
status_line = status_df[['status immediate independence',
                         'status immediate unification',
                         'status is already an independent country',
                         'status is already an unified country',
                         'status maintain the status quo']].copy()

# line chart
status_line.plot.line(marker = 'o', figsize=(10,4))
plt.xlabel("Survey No.")
plt.ylabel("Ethnic Identification (%)")
plt.legend(bbox_to_anchor=(1,1), loc="upper left")
plt.title("TAIWANESE STATUS");
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [1], in <cell line: 2>()
      1 # Status
----> 2 status_line = status_df[['status immediate independence',
      3                          'status immediate unification',
      4                          'status is already an independent country',
      5                          'status is already an unified country',
      6                          'status maintain the status quo']].copy()
      8 # line chart
      9 status_line.plot.line(marker = 'o', figsize=(10,4))

NameError: name 'status_df' is not defined
```

The final line chart shows how respondents feel about Taiwanese-China status. They selected a range of responses from immediate independence or already independent to immediate unification or already unified. The rest of responses are variations of maintaining the unclear status quo. In each survey, a plurality of respondents want to maintain the status quo (except in survey one). The second most common response in most surveys is immediate independence, which becomes more popular in the final year of surveys. Regardless of the survey, responses related to unification seem quite unpopular.

Visualizations: Maps

Next we download and read in spatial data from The University of Texas at Austin's Geo-Library in the form of a geojson file to help visualise our dependent variables from TEDS by top level administrative divisons in Taiwan, which include: counties, special municipalities and autonomous municipalities (UT-Austin, 2015).

The package *geopandas* is a geospatial extension of the pandas package, allowing data frames to be represented spatially by interacting with matplotlib to plot maps 1-8 (Jordahl et al., 2020). This package allows geojson files to be read into Python, as well as geometric objects to represented in data frames under the **geometry** variable column. Using this package and its' interaction with matplotlib allows for choropleth maps to be produced.

The geospatial techniques used in this section from *geopandas* were adapted from Yoh Kawano, a computational scientist at UCLAs, Github respository (Kawano, 2021).

We create two types of maps for each dependent variable:

1. Maps that show the percentage change in the dependent variable before and after the Hong Kong Protests, and
2. Maps that show the dependent variable's average accross all periods.

For class variables, where there is no clear ordering (i.e. identification as Taiwanese or Chinese) we include maps for several categories.

In [17]:
```
# reads in the geojson file as a geo-data frame
geodf = gpd.read_file("Data/stanford-fn648mm8787-geojson.json")
```

We then clean the geojson file into something more usable.

In [18]:
```
# renames columns into our own variable names
geodf = geodf[["id", "varname_2", "geometry"]]
geodf.columns = ["loc_id", "name", "geometry"]
```

In [19]:
```
# replaces useless information and converts data to lower case
geodf["loc_id"] = geodf["loc_id"].str.replace("fn648mm8787.", "")
geodf["name"] = geodf["name"].str.lower()
```

```
C:\Users\19372\AppData\Local\Temp\ipykernel_1616\1267108561.py:2: FutureWarning: The default value of regex wil
l change from True to False in a future version.
  geodf["loc_id"] = geodf["loc_id"].str.replace("fn648mm8787.", "")
```

We then prepare to match the geo-data frame to our data from TEDs surveys, this allows for our data from TEDS to later be represented spatially.

In [20]:
```
# prepares our TEDS to become a geo-data frame by asigning a new name
geodf_survey = df_survey
```

In [21]:
```
# we assign a new column based on our existing location column
geodf_survey["loc_id"] = geodf_survey["location"]
```

In [22]:
```
# we then replace the string values in our "loc_id" with the numerical values for each string in the geojson fi
# by comparing "location" column in geodf_survey to "name" column in geodf
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["kinmen county"], "1")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["lienchiang county"], "2")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["hsinchu city"], "3")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["kaohsiung city"], "4")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["new taipei city"], "5")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["taichung city"], "6")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["tainan city"], "7")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["taipei city"], "8")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["taoyuan city"], "9")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["changhua county"], "10")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["chiayi city"], "11")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["chiayi county"], "12")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["hsinchu county"], "13")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["hualien county"], "14")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["keelung city"], "15")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["miaoli county"], "16")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["nantou county"], "17")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["penghu county"], "18")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["pingtung county"], "19")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["taitung county"], "20")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["yilan county"], "21")
geodf_survey["loc_id"] = geodf_survey["loc_id"].replace(["yunlin county"], "22")
```

In order to measure the change in attitudes on our relevant dependent variables we break our new geo-data frame into a pre- and post-period subset of surveys, where the pre-period is all the TEDs survey values from 2017 and 2018 and the post-period values from 2019-2020 which are after the start of the Hong Kong Protests.

In [23]:
```
# creates a new data frame of pre Hong Kong protests grouping
geodf_filter = geodf_survey["survey"] <= 11
geodf_pre = geodf_survey[geodf_filter]
#geodf_pre

# creates a new data frame of post Hong Kong protests grouping
geodf_filter = geodf_survey["survey"] > 11
geodf_post = geodf_survey[geodf_filter]
```

Tsai's Cross-Strait Relations Percent Change

In order to measure the difference in Tsai Ing-Wen's handling of Cross-Strait Relations with China we use the **tsai_csr** variable and measure its percent change between the pre- and post-period in each top level administrative divison in Taiwan.

In [24]:
```
# first we remove the "not specified" variable from our analysis (NaN variables are already ignored)
geodf_filter = geodf_pre["tsai_csr"] != "not specified"
geodf_pre_csr = geodf_pre[geodf_filter]
#geodf_pre_csr

geodf_filter = geodf_post["tsai_csr"] != "not specified"
geodf_post_csr = geodf_post[geodf_filter]
#geodf_post_csr

# we then assign dummy variables for all possible values per top level administrative division in taiwan
```

```
# first for pre-period
dummies_csr = pd.get_dummies(geodf_pre_csr["tsai_csr"])
geodf_pre_csr = geodf_pre_csr.join(dummies_csr)
geodf_pre_csr["tsai_satisfied"] = geodf_pre_csr["satisfied"] + geodf_pre_csr["very satisfied"]
#geodf_pre_csr

# then for post-period
dummies_csr = pd.get_dummies(geodf_post_csr["tsai_csr"])
geodf_post_csr = geodf_post_csr.join(dummies_csr)
geodf_post_csr["tsai_satisfied"] = geodf_post_csr["satisfied"] + geodf_post_csr["very satisfied"]
#geodf_post_csr
```

In [25]:
```
# first for pre-period
# we then use the groupby function to count the number of variable observations and the total number of observa
sat_num = geodf_pre_csr.groupby(["loc_id"])[["tsai_satisfied"]].sum()
loc_num = geodf_pre_csr.groupby(["loc_id"])[["loc_id"]].count()
csr_num_pre = pd.concat([sat_num, loc_num], axis=1)
#csr_num_pre

# convert to percent terms
csr_num_pre["csr_sat_rate"] = csr_num_pre["tsai_satisfied"] / csr_num_pre["loc_id"] * 100
#csr_num_pre

# drop duplicate data
csr_num_pre = csr_num_pre.drop(["tsai_satisfied"], axis=1)
csr_num_pre = csr_num_pre.drop(["loc_id"], axis=1)
#csr_num_pre
```

In [26]:
```
# then for post-period
# we then use the groupby function to count the number of variable observations and the total number of observa
sat_num = geodf_post_csr.groupby(["loc_id"])[["tsai_satisfied"]].sum()
loc_num = geodf_post_csr.groupby(["loc_id"])[["loc_id"]].count()
csr_num_post = pd.concat([sat_num, loc_num], axis=1)
#csr_num_post

# convert to percent terms
csr_num_post["csr_sat_rate"] = csr_num_post["tsai_satisfied"] / csr_num_post["loc_id"] * 100
#csr_num_post

# drop duplicate data
csr_num_post = csr_num_post.drop(["tsai_satisfied"], axis=1)
csr_num_post = csr_num_post.drop(["loc_id"], axis=1)
#csr_num_post
```

In [27]:
```
# calculate change in rate
csr_num = csr_num_pre
csr_num["csr_change"] = csr_num_post["csr_sat_rate"] - csr_num_pre["csr_sat_rate"]

# drop duplicate data
csr_num = csr_num.drop(["csr_sat_rate"], axis = 1)
#csr_num
```

In [28]:
```
# add data to geo-data frame
geodf_survey = pd.merge(geodf_survey, csr_num, left_on="loc_id", right_on="loc_id")
#geodf_survey
```

Tsai's Cross-Strait Relations Policy Average

We also wanted to measure Tsai Ing-Wen's baseline approval rating of her handling of Cross-Strait Relations across all surveys.

In [29]:
```
# first we remove the "not specified" variable from our analysis (NaN variables are already ignored)
geodf_filter = geodf_survey["tsai_csr"] != "not specified"
geodf_csr = geodf_survey[geodf_filter]

# we then assign dummy variables for all possible values per top level administrative division in taiwan
dummies_csr = pd.get_dummies(geodf_csr["tsai_csr"])
geodf_csr = geodf_csr.join(dummies_csr)
geodf_csr["tsai_satisfied"] = geodf_csr["satisfied"] + geodf_csr["very satisfied"]
#geodf_csr
```

In [30]:
```
# we then use the groupby function to count the number of variable observations and the total number of observa
sat_num = geodf_csr.groupby(["loc_id"])[["tsai_satisfied"]].sum()
loc_num = geodf_csr.groupby(["loc_id"])[["loc_id"]].count()
csr_num = pd.concat([sat_num, loc_num], axis=1)

# convert to percent terms
csr_num["csr_average"] = csr_num["tsai_satisfied"] / csr_num["loc_id"] * 100

# drop duplicate data
csr_num = csr_num.drop(["tsai_satisfied"], axis=1)
csr_num = csr_num.drop(["loc_id"], axis=1)

# add data to geo-data frame
geodf_survey = pd.merge(geodf_survey, csr_num, left_on="loc_id", right_on="loc_id")
```

Cross-Strait Relations Most Important Percent Change

To measure the difference in importance of Cross-Strait Relations with China we use the **most_import** variable and measure its percent change between the pre- and post-period in each top level administrative divison in Taiwan.

In [31]:
```
# first we remove the "not specified" variable from our analysis (NaN variables are already ignored)
geodf_filter = geodf_pre["most_import"] != "not specified"
geodf_pre_import = geodf_pre[geodf_filter]
#geodf_pre_import

geodf_filter = geodf_post["most_import"] != "not specified"
geodf_post_import = geodf_post[geodf_filter]
#geodf_post_import

# we then assign dummy variables for all possible values per top level administrative division in taiwan
# first for pre-period
dummies_import = pd.get_dummies(geodf_pre_import["most_import"])
geodf_pre_import = geodf_pre_import.join(dummies_import)
#geodf_pre_import

# then for post-period
dummies_import = pd.get_dummies(geodf_post_import["most_import"])
geodf_post_import = geodf_post_import.join(dummies_import)
#geodf_post_import
```

In [32]:
```
# first for pre-period
# we then use the groupby function to count the number of variable observations and the total number of observa
most_num = geodf_pre_import.groupby(["loc_id"])[["cross-strait relations"]].sum()
loc_num = geodf_pre_import.groupby(["loc_id"])[["loc_id"]].count()
import_num_pre = pd.concat([most_num, loc_num], axis=1)

# convert to percent terms
import_num_pre["csr_import_rate"] = import_num_pre["cross-strait relations"] / import_num_pre["loc_id"] * 100

# drop duplicate data
import_num_pre = import_num_pre.drop(["cross-strait relations"], axis=1)
import_num_pre = import_num_pre.drop(["loc_id"], axis=1)
#import_num_pre
```

In [33]:
```
# then for post-period
# we then use the groupby function to count the number of variable observations and the total number of observa
most_num = geodf_post_import.groupby(["loc_id"])[["cross-strait relations"]].sum()
loc_num = geodf_post_import.groupby(["loc_id"])[["loc_id"]].count()
import_num_post = pd.concat([most_num, loc_num], axis=1)

# convert to percent terms
import_num_post["csr_import_rate"] = import_num_post["cross-strait relations"] / import_num_post["loc_id"] * 10

# drop duplicate data
import_num_post = import_num_post.drop(["cross-strait relations"], axis=1)
import_num_post = import_num_post.drop(["loc_id"], axis=1)
#import_num_post
```

In [34]:
```
# calculate change in rate
import_num = import_num_pre
import_num["import_change"] = import_num_post["csr_import_rate"] - import_num_pre["csr_import_rate"]

# drop duplicate data
import_num = import_num.drop(["csr_import_rate"], axis = 1)
#import_num
```

In [35]:
```
# add data to geo-data frame
geodf_survey = pd.merge(geodf_survey, import_num, left_on="loc_id", right_on="loc_id")
#geodf_survey
```

Cross-Strait Relations Policy Most Important Average

We also measure how important Cross-Strait relations are across all surveys as follows:

In [36]:
```
# first we remove the "not specified" variable from our analysis (NaN variables are already ignored)
geodf_filter = geodf_survey["most_import"] != "not specified"
geodf_import = geodf_survey[geodf_filter]

# we then assign dummy variables for all possible values per top level administrative division in taiwan
dummies_import = pd.get_dummies(geodf_import["most_import"])
geodf_import = geodf_import.join(dummies_import)
#geodf_import
```

In [37]:
```
# we then use the groupby function to count the number of variable observations and the total number of observa
most_num = geodf_pre_import.groupby(["loc_id"])[["cross-strait relations"]].sum()
loc_num = geodf_pre_import.groupby(["loc_id"])[["loc_id"]].count()
import_num = pd.concat([most_num, loc_num], axis=1)
```

```
# convert to percent terms
import_num["import_average"] = import_num["cross-strait relations"] / import_num["loc_id"] * 100

# drop duplicate data
import_num = import_num.drop(["cross-strait relations"], axis=1)
import_num = import_num.drop(["loc_id"], axis=1)

# add data to geo-data frame
geodf_survey = pd.merge(geodf_survey, import_num, left_on="loc_id", right_on="loc_id")
```

Identity Percent Change

For measuring the difference in how people in Taiwan identify ethnically, we use the **identity** variable and measure its percent change between the pre- and post-period in each top level administrative divison in Taiwan.

In [38]:
```
# first we remove the "not specified" variable from our analysis (NaN variables are already ignored)
geodf_filter = geodf_pre["identity"] != "not specified"
geodf_pre_identity = geodf_pre[geodf_filter]
#geodf_pre_identity

geodf_filter = geodf_post["identity"] != "not specified"
geodf_post_identity = geodf_post[geodf_filter]
#geodf_post_identity

# we then assign dummy variables for all possible values per top level administrative division in taiwan
# first for pre-period
dummies_identity = pd.get_dummies(geodf_pre_identity["identity"])
geodf_pre_identity = geodf_pre_identity.join(dummies_identity)
#geodf_pre_identity

# then for post-period
dummies_identity = pd.get_dummies(geodf_post_identity["identity"])
geodf_post_identity = geodf_post_identity.join(dummies_identity)
#geodf_post_identity
```

A. Taiwanese

First, we measure the change in identifying as **taiwanese** between the pre- and post-period.

In [39]:
```
# first for pre-period
# we then use the groupby function to count the number of variable observations and the total number of observa
tw_num = geodf_pre_identity.groupby(["loc_id"])[["taiwanese"]].sum()
loc_num = geodf_pre_identity.groupby(["loc_id"])[["loc_id"]].count()
identity_num_pre = pd.concat([tw_num, loc_num], axis=1)

# convert to percent terms
identity_num_pre["identity_rate"] = identity_num_pre["taiwanese"] / identity_num_pre["loc_id"] * 100

# drop duplicate data
identity_num_pre = identity_num_pre.drop(["taiwanese"], axis=1)
identity_num_pre = identity_num_pre.drop(["loc_id"], axis=1)
#identity_num_pre
```

In [40]:
```
# then for post-period
# we then use the groupby function to count the number of variable observations and the total number of observa
tw_num = geodf_post_identity.groupby(["loc_id"])[["taiwanese"]].sum()
loc_num = geodf_post_identity.groupby(["loc_id"])[["loc_id"]].count()
identity_num_post = pd.concat([tw_num, loc_num], axis=1)

# convert to percent terms
identity_num_post["identity_rate"] = identity_num_post["taiwanese"] / identity_num_post["loc_id"] * 100

# drop duplicate data
identity_num_post = identity_num_post.drop(["taiwanese"], axis=1)
identity_num_post = identity_num_post.drop(["loc_id"], axis=1)
#identity_num_post
```

In [41]:
```
# calculate change in rate
identity_num = identity_num_pre
identity_num["taiwanese_change"] = identity_num_post["identity_rate"] - identity_num_pre["identity_rate"]

# drop duplicate data
identity_num = identity_num.drop(["identity_rate"], axis = 1)
#identity_num
```

In [42]:
```
# add data to geo-data frame
geodf_survey = pd.merge(geodf_survey, identity_num, left_on="loc_id", right_on="loc_id")
#geodf_survey
```

B. Both

Next, we measure the change in identifying as **both** (Taiwanese and Chinese) between the pre- and post-period.

In [43]:
```
# first for pre-period
# we then use the groupby function to count the number of variable observations and the total number of observa
tw_num = geodf_pre_identity.groupby(["loc_id"])[["both"]].sum()
loc_num = geodf_pre_identity.groupby(["loc_id"])[["loc_id"]].count()
identity_num_pre = pd.concat([tw_num, loc_num], axis=1)

# convert to percent terms
identity_num_pre["identity_rate"] = identity_num_pre["both"] / identity_num_pre["loc_id"] * 100

# drop duplicate data
identity_num_pre = identity_num_pre.drop(["both"], axis=1)
identity_num_pre = identity_num_pre.drop(["loc_id"], axis=1)
#identity_num_pre
```

In [44]:
```
# then for post-period
# we then use the groupby function to count the number of variable observations and the total number of observa
tw_num = geodf_post_identity.groupby(["loc_id"])[["both"]].sum()
loc_num = geodf_post_identity.groupby(["loc_id"])[["loc_id"]].count()
identity_num_post = pd.concat([tw_num, loc_num], axis=1)

# convert to percent terms
identity_num_post["identity_rate"] = identity_num_post["both"] / identity_num_post["loc_id"] * 100

# drop duplicate data
identity_num_post = identity_num_post.drop(["both"], axis=1)
identity_num_post = identity_num_post.drop(["loc_id"], axis=1)
#identity_num_post
```

In [45]:
```
# calculate change in rate
identity_num = identity_num_pre
identity_num["both_change"] = identity_num_post["identity_rate"] - identity_num_pre["identity_rate"]

# drop duplicate data
identity_num = identity_num.drop(["identity_rate"], axis = 1)
#identity_num
```

In [46]:
```
# add data to geo-data frame
geodf_survey = pd.merge(geodf_survey, identity_num, left_on="loc_id", right_on="loc_id")
#geodf_survey
```

C. Chinese

Finally, we measure the change in identifying as **chinese** between the pre- and post-period.

In [47]:
```
# first for pre-period
# we then use the groupby function to count the number of variable observations and the total number of observa
tw_num = geodf_pre_identity.groupby(["loc_id"])[["chinese"]].sum()
loc_num = geodf_pre_identity.groupby(["loc_id"])[["loc_id"]].count()
identity_num_pre = pd.concat([tw_num, loc_num], axis=1)

# convert to percent terms
identity_num_pre["identity_rate"] = identity_num_pre["chinese"] / identity_num_pre["loc_id"] * 100

identity_num_pre = identity_num_pre.drop(["chinese"], axis=1)
identity_num_pre = identity_num_pre.drop(["loc_id"], axis=1)
#identity_num_pre
```

In [48]:
```
# then for post-period
# we then use the groupby function to count the number of variable observations and the total number of observa
tw_num = geodf_post_identity.groupby(["loc_id"])[["chinese"]].sum()
loc_num = geodf_post_identity.groupby(["loc_id"])[["loc_id"]].count()
identity_num_post = pd.concat([tw_num, loc_num], axis=1)

# convert to percent terms
identity_num_post["identity_rate"] = identity_num_post["chinese"] / identity_num_post["loc_id"] * 100

# drop duplicate data
identity_num_post = identity_num_post.drop(["chinese"], axis=1)
identity_num_post = identity_num_post.drop(["loc_id"], axis=1)
#identity_num_post
```

In [49]:
```
# calculate change in rate
identity_num = identity_num_pre
identity_num["chinese_change"] = identity_num_post["identity_rate"] - identity_num_pre["identity_rate"]

# drop duplicate data
identity_num = identity_num.drop(["identity_rate"], axis = 1)
#identity_num
```

In [50]:
```
# add data to geo-data frame
geodf_survey = pd.merge(geodf_survey, identity_num, left_on="loc_id", right_on="loc_id")
#geodf_survey
```

Identity Average

Identity Average

It is also important to measure the baseline **taiwanese**, **both** and **chinese** identities in percent terms across all surveys per administrative division.

In [51]:
```
# first we remove the "not specified" variable from our analysis (NaN variables are already ignored)
geodf_filter = geodf_survey["identity"] != "not specified"
geodf_identity = geodf_survey[geodf_filter]

# we then assign dummy variables for all possible values per top level administrative division in taiwan
dummies_identity = pd.get_dummies(geodf_identity["identity"])
geodf_identity = geodf_identity.join(dummies_identity)
#geodf_identity
```

In [52]:
```
# we then use the groupby function to count the number of variable observations and the total number of observa
tw_num = geodf_identity.groupby(["loc_id"])[["taiwanese"]].sum()
loc_num = geodf_identity.groupby(["loc_id"])[["loc_id"]].count()
identity_num = pd.concat([tw_num, loc_num], axis=1)

# convert to percent terms
identity_num["taiwanese_average"] = identity_num["taiwanese"] / identity_num["loc_id"] * 100

# drop duplicate data
identity_num = identity_num.drop(["taiwanese"], axis=1)
identity_num = identity_num.drop(["loc_id"], axis=1)

# add data to geo-data frame
geodf_survey = pd.merge(geodf_survey, identity_num, left_on="loc_id", right_on="loc_id")

# we then use the groupby function to count the number of variable observations and the total number of observa
tw_num = geodf_identity.groupby(["loc_id"])[["both"]].sum()
loc_num = geodf_identity.groupby(["loc_id"])[["loc_id"]].count()
identity_num = pd.concat([tw_num, loc_num], axis=1)

# convert to percent terms
identity_num["both_average"] = identity_num["both"] / identity_num["loc_id"] * 100

# drop duplicate data
identity_num = identity_num.drop(["both"], axis=1)
identity_num = identity_num.drop(["loc_id"], axis=1)

# add data to geo-data frame
geodf_survey = pd.merge(geodf_survey, identity_num, left_on="loc_id", right_on="loc_id")

# we then use the groupby function to count the number of variable observations and the total number of observa
tw_num = geodf_identity.groupby(["loc_id"])[["chinese"]].sum()
loc_num = geodf_identity.groupby(["loc_id"])[["loc_id"]].count()
identity_num = pd.concat([tw_num, loc_num], axis=1)

# convert to percent terms
identity_num["chinese_average"] = identity_num["chinese"] / identity_num["loc_id"] * 100

# drop duplicate data
identity_num = identity_num.drop(["chinese"], axis=1)
identity_num = identity_num.drop(["loc_id"], axis=1)

# add data to geo-data frame
geodf_survey = pd.merge(geodf_survey, identity_num, left_on="loc_id", right_on="loc_id")
```

Status Percentage Change

The difference in how people in Taiwan view its political status is measured using the **status** and its percent change between the pre- and post-period in each top level administrative divison in Taiwan.

In [53]:
```
# first we remove the "not specified" variable from our analysis (NaN variables are already ignored)
geodf_filter = geodf_pre["status"] != "not specified"
geodf_pre_status = geodf_pre[geodf_filter]
#geodf_pre_status

geodf_filter = geodf_post["status"] != "not specified"
geodf_post_status = geodf_post[geodf_filter]
#geodf_post_status

# we then assign dummy variables for all possible values per top level administrative division in taiwan
# first for pre-period
dummies_status = pd.get_dummies(geodf_pre_status["status"])
geodf_pre_status = geodf_pre_status.join(dummies_status)
geodf_pre_status["ind_status"] = geodf_pre_status["immediate independence"] + geodf_pre_status["is already an i
geodf_pre_status["msq_status"] = (geodf_pre_status["maintain the status quo, decide either unification or indep
                                  + geodf_pre_status["maintain the status quo forever"]
                                  + geodf_pre_status["maintain the status quo, move toward independence in the f
                                  + geodf_pre_status["maintain the status quo, move toward unification in the fu
geodf_pre_status["uni_status"] = geodf_pre_status["immediate unification"] + geodf_pre_status["is already an un
#geodf_pre_status
```

```
# then for post-period
dummies_status = pd.get_dummies(geodf_post_status["status"])
geodf_post_status = geodf_post_status.join(dummies_status)
geodf_post_status["ind_status"] = geodf_post_status["immediate independence"] + geodf_post_status["is already a
geodf_post_status["msq_status"] = (geodf_post_status["maintain the status quo, decide either unification or ind
                                   + geodf_post_status["maintain the status quo forever"]
                                   + geodf_post_status ["maintain the status quo, move toward independence in th
                                   + geodf_post_status["maintain the status quo, move toward unification in the
geodf_post_status["uni_status"] = geodf_post_status["immediate unification"] #+ geodf_post_status["is already a
#geodf_post_status
```

A. Independence

First, we measure the change in average support for independence in the **status** variable between the pre- and post-period.

In [54]:
```
# first for pre-period
# we then use the groupby function to count the number of variable observations and the total number of observa
id_num = geodf_pre_status.groupby(["loc_id"])[["ind_status"]].sum()
loc_num = geodf_pre_status.groupby(["loc_id"])[["loc_id"]].count()
status_num_pre = pd.concat([id_num, loc_num], axis=1)

# convert to percent terms
status_num_pre["status_rate"] = status_num_pre["ind_status"] / status_num_pre["loc_id"] * 100

# drop duplicate data
status_num_pre = status_num_pre.drop(["ind_status"], axis=1)
status_num_pre = status_num_pre.drop(["loc_id"], axis=1)
#status_num_pre
```

In [55]:
```
# then for post-period
# we then use the groupby function to count the number of variable observations and the total number of observa
id_num = geodf_post_status.groupby(["loc_id"])[["ind_status"]].sum()
loc_num = geodf_post_status.groupby(["loc_id"])[["loc_id"]].count()
status_num_post = pd.concat([id_num, loc_num], axis=1)

# convert to percent terms
status_num_post["status_rate"] = status_num_post["ind_status"] / status_num_post["loc_id"] * 100

# drop duplicate data
status_num_post = status_num_post.drop(["ind_status"], axis=1)
status_num_post = status_num_post.drop(["loc_id"], axis=1)
#status_num_post
```

In [56]:
```
# calculate change in rate
status_num = status_num_pre
status_num["ind_change"] = status_num_post["status_rate"] - status_num_pre["status_rate"]

# drop duplicate data
status_num = status_num.drop(["status_rate"], axis = 1)
#status_num
```

In [57]:
```
# add data to geo-data frame
geodf_survey = pd.merge(geodf_survey, status_num, left_on="loc_id", right_on="loc_id")
#geodf_survey
```

B. Maintain Status Quo

Next, we measure the change in average support for maintaining the status quo in the **status** variable between the pre- and post-period.

In [58]:
```
# first for pre-period
# we then use the groupby function to count the number of variable observations and the total number of observa
id_num = geodf_pre_status.groupby(["loc_id"])[["msq_status"]].sum()
loc_num = geodf_pre_status.groupby(["loc_id"])[["loc_id"]].count()
status_num_pre = pd.concat([id_num, loc_num], axis=1)

# convert to percent terms
status_num_pre["status_rate"] = status_num_pre["msq_status"] / status_num_pre["loc_id"] * 100

# drop duplicate data
status_num_pre = status_num_pre.drop(["msq_status"], axis=1)
status_num_pre = status_num_pre.drop(["loc_id"], axis=1)
#status_num_pre
```

In [59]:
```
# then for post-period
# we then use the groupby function to count the number of variable observations and the total number of observa
id_num = geodf_post_status.groupby(["loc_id"])[["msq_status"]].sum()
loc_num = geodf_post_status.groupby(["loc_id"])[["loc_id"]].count()
status_num_post = pd.concat([id_num, loc_num], axis=1)

# convert to percent terms
status_num_post["status_rate"] = status_num_post["msq_status"] / status_num_post["loc_id"] * 100

# drop duplicate data
status_num_post = status_num_post.drop(["msq_status"], axis=1)
```

```
status_num_post = status_num_post.drop(["loc_id"], axis=1)
#status_num_post
```

In [60]:
```
# calculate change in rate
status_num = status_num_pre
status_num["msq_change"] = status_num_post["status_rate"] - status_num_pre["status_rate"]

# drop duplicate data
status_num = status_num.drop(["status_rate"], axis = 1)
#status_num
```

In [61]:
```
# add data to geo-data frame
geodf_survey = pd.merge(geodf_survey, status_num, left_on="loc_id", right_on="loc_id")
#geodf_survey
```

C. Unification

Finally, we measure the change in average support for unification in the **status** variable between the pre- and post-period.

In [62]:
```
# first for pre-period
# we then use the groupby function to count the number of variable observations and the total number of observa
id_num = geodf_pre_status.groupby(["loc_id"])[["uni_status"]].sum()
loc_num = geodf_pre_status.groupby(["loc_id"])[["loc_id"]].count()
status_num_pre = pd.concat([id_num, loc_num], axis=1)

# convert to percent terms
status_num_pre["status_rate"] = status_num_pre["uni_status"] / status_num_pre["loc_id"] * 100

# drop duplicate data
status_num_pre = status_num_pre.drop(["uni_status"], axis=1)
status_num_pre = status_num_pre.drop(["loc_id"], axis=1)
#status_num_pre
```

In [63]:
```
# then for post-period
# we then use the groupby function to count the number of variable observations and the total number of observa
id_num = geodf_post_status.groupby(["loc_id"])[["uni_status"]].sum()
loc_num = geodf_post_status.groupby(["loc_id"])[["loc_id"]].count()
status_num_post = pd.concat([id_num, loc_num], axis=1)

# convert to percent terms
status_num_post["status_rate"] = status_num_post["uni_status"] / status_num_post["loc_id"] * 100

# drop duplicate data
status_num_post = status_num_post.drop(["uni_status"], axis=1)
status_num_post = status_num_post.drop(["loc_id"], axis=1)
#status_num_post
```

In [64]:
```
# calculate change in rate
status_num = status_num_pre
status_num["uni_change"] = status_num_post["status_rate"] - status_num_pre["status_rate"]

# drop duplicate data
status_num = status_num.drop(["status_rate"], axis = 1)
#status_num
```

In [65]:
```
# add data to geo-data frame
geodf_survey = pd.merge(geodf_survey, status_num, left_on="loc_id", right_on="loc_id")
#geodf_survey
```

Status Average

It was also important to measure the baseline average **status** level in percent terms across all surveys by calculating it below.

In [66]:
```
# first we remove the "not specified" variable from our analysis (NaN variables are already ignored)
geodf_filter = geodf_survey["status"] != "not specified"
geodf_status = geodf_survey[geodf_filter]

# we then assign dummy variables for all possible values per top level administrative division in taiwan
dummies_status = pd.get_dummies(geodf_status["status"])
geodf_status = geodf_status.join(dummies_status)
geodf_status["ind_status"] = geodf_status["immediate independence"] + geodf_status["is already an independent c
geodf_status["msq_status"] = (geodf_status["maintain the status quo, decide either unification or independence
                              + geodf_status["maintain the status quo forever"]
                              + geodf_status["maintain the status quo, move toward independence in the futur
                              + geodf_status["maintain the status quo, move toward unification in the future
geodf_status["uni_status"] = geodf_status["immediate unification"] + geodf_status["is already an unified countr
#geodf_status
```

In [67]:
```
# we then use the groupby function to count the number of variable observations and the total number of observa
id_num = geodf_status.groupby(["loc_id"])[["ind_status"]].sum()
loc_num = geodf_status.groupby(["loc_id"])[["loc_id"]].count()
status_num = pd.concat([id_num, loc_num], axis=1)
```

```
# convert to percent terms
status_num["ind_average"] = status_num["ind_status"] / status_num["loc_id"] * 100

# drop duplicate data
status_num = status_num.drop(["ind_status"], axis=1)
status_num = status_num.drop(["loc_id"], axis=1)

# add data to geo-data frame
geodf_survey = pd.merge(geodf_survey, status_num, left_on="loc_id", right_on="loc_id")

# we then use the groupby function to count the number of variable observations and the total number of observa
id_num = geodf_status.groupby(["loc_id"])[["msq_status"]].sum()
loc_num = geodf_status.groupby(["loc_id"])[["loc_id"]].count()
status_num = pd.concat([id_num, loc_num], axis=1)

# convert to percent terms
status_num["msq_average"] = status_num["msq_status"] / status_num["loc_id"] * 100

# drop duplicate data
status_num = status_num.drop(["msq_status"], axis=1)
status_num = status_num.drop(["loc_id"], axis=1)

# add data to geo-data frame
geodf_survey = pd.merge(geodf_survey, status_num, left_on="loc_id", right_on="loc_id")

# we then use the groupby function to count the number of variable observations and the total number of observa
id_num = geodf_status.groupby(["loc_id"])[["uni_status"]].sum()
loc_num = geodf_status.groupby(["loc_id"])[["loc_id"]].count()
status_num = pd.concat([id_num, loc_num], axis=1)

# convert to percent terms
status_num["uni_average"] = status_num["uni_status"] / status_num["loc_id"] * 100

# drop duplicate data
status_num = status_num.drop(["uni_status"], axis=1)
status_num = status_num.drop(["loc_id"], axis=1)

# add data to geo-data frame
geodf_survey = pd.merge(geodf_survey, status_num, left_on="loc_id", right_on="loc_id")

#geodf_survey
```

Map Visualization Results

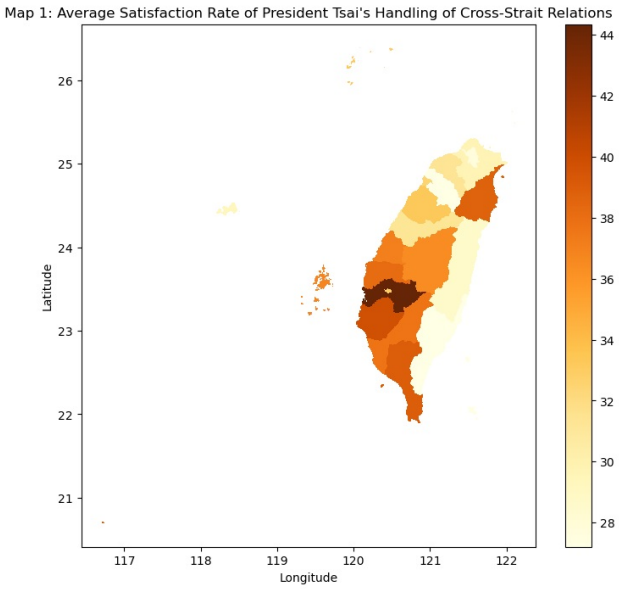
Now that we have calculated all of our dependent variables per top-level administrative division in Taiwan, we next display them visually below, by first merging our TEDS survey data with our geojson data as follows:

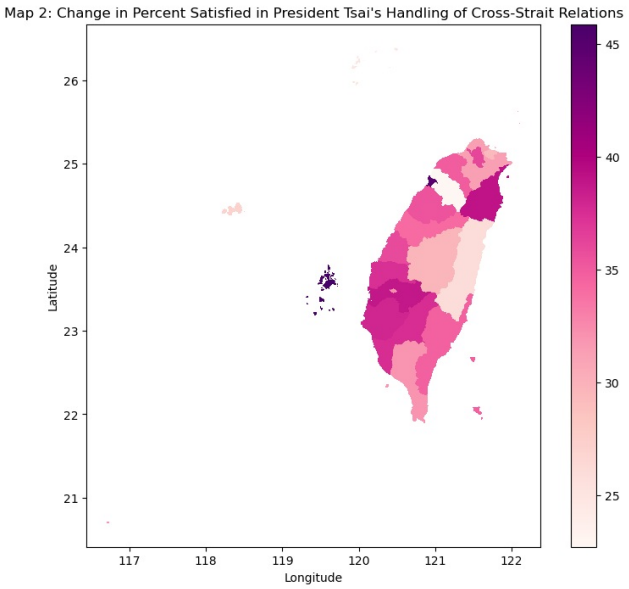
In [68]:
```
# creates a spatial join of our two data frames
geodf_join = pd.merge(geodf, geodf_survey, left_on="loc_id", right_on="loc_id")
```

First, we plot **Map 1** and **Map 2** of change in satisfaction of President Tsai's handling of Cross-Strait Relations, and the baseline levels as well respectively:

In [69]:
```
# set plot bounds
fig,ax = plt.subplots(ncols = 2, figsize = (20, 8))
ax1, ax2 = ax

# plot variable of map 1
geodf_join.plot(ax = ax1, column = "csr_average", cmap = "YlOrBr", legend = True)
ax1.set_title("Map 1: Average Satisfaction Rate of President Tsai's Handling of Cross-Strait Relations", fontsi
ax1.set_xlabel("Longitude", fontsize = 10)
ax1.set_ylabel("Latitude", fontsize = 10)

# plot vairable of map 2
geodf_join.plot(ax = ax2, column = "csr_change", cmap = "RdPu", legend = True)
ax2.set_title("Map 2: Change in Percent Satisfied in President Tsai's Handling of Cross-Strait Relations", font
ax2.set_xlabel("Longitude", fontsize = 10)
ax2.set_ylabel("Latitude", fontsize = 10)
```

Out[69]:
```
Text(1090.0697650950895, 0.5, 'Latitude')
```



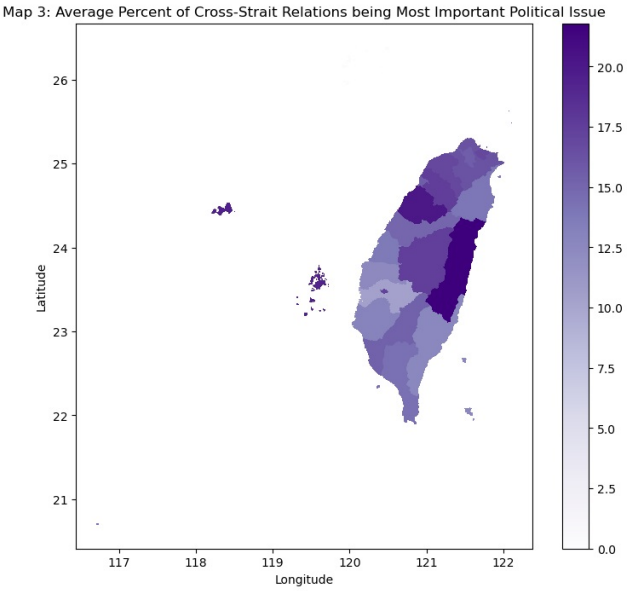


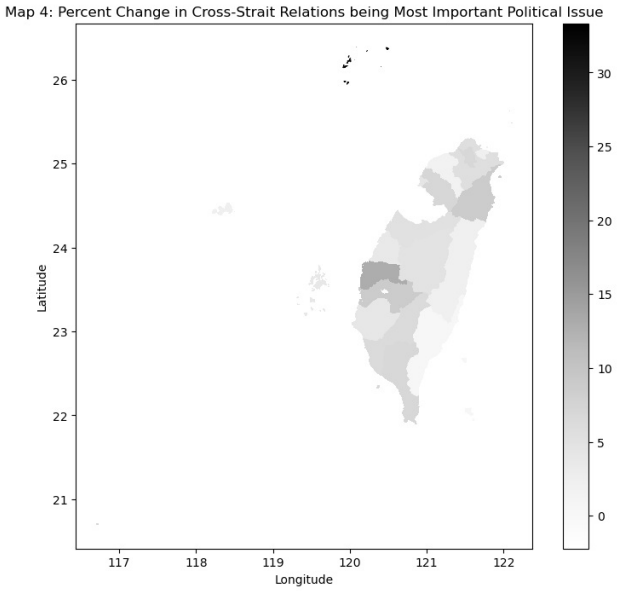
From **Map 1** above, we can see that over the entire period of our surveys, she never had an average satisfaction rate over 50% on this issue. Moreover, where she has a strong plurality of support in **Map 1** comes from the Southwest of Taiwan where she recieved her strong vote share in the 2016 election (Taiwanese Government, 2016). However, when we look at **Map 2** we can see that there are very large swings in support of her handling in Cross-Strait Relations after the Hong Kong protests, suggesting that she recieved a strong wind of support in her handling of the issue during the 2019-2020 period. While it is impossible to draw a causal relationship to the Hong Kong protests, the change in support does correlate strongly with this time period difference.

Second, we choose to plot our second dependent variable, **most_import**, to display spatially the relevance of Cross-Strait Relations to political life for people who live in Taiwan in **Map 3** and **Map 4**.

In [70]:
```
# set plot bounds
fig,ax = plt.subplots(ncols = 2, figsize = (20, 8))
ax1, ax2 = ax

# plot variable of map 3
geodf_join.plot(ax = ax1, column = "import_average", cmap = "Purples", legend = True)
ax1.set_title("Map 3: Average Percent of Cross-Strait Relations being Most Important Political Issue", fontsize
ax1.set_xlabel("Longitude", fontsize = 10)
ax1.set_ylabel("Latitude", fontsize = 10)

# plot variable of map 4
geodf_join.plot(ax = ax2, column = "import_change", cmap = "Greys", legend = True)
ax2.set_title("Map 4: Percent Change in Cross-Strait Relations being Most Important Political Issue", fontsize
ax2.set_xlabel("Longitude", fontsize = 10)
ax2.set_ylabel("Latitude", fontsize = 10)
```

Out[70]:
```
Text(1090.0697650950895, 0.5, 'Latitude')
```



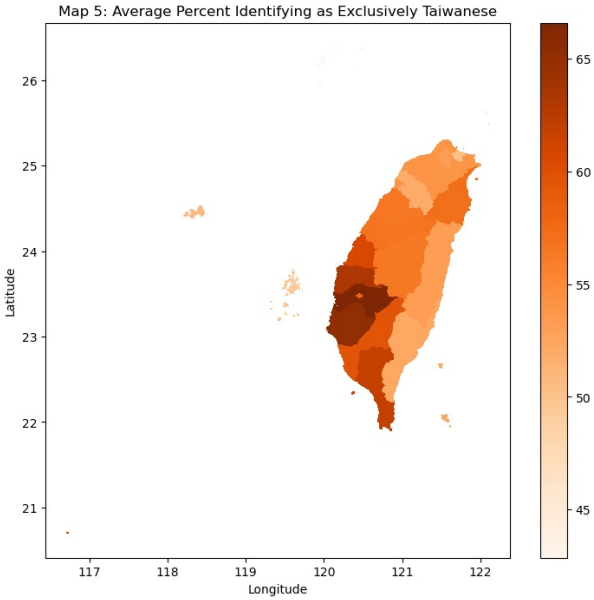


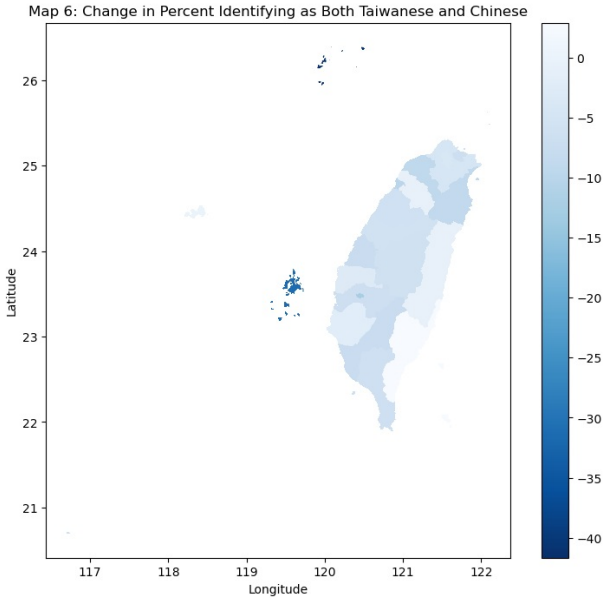
As we can see above in **Map 3**, Cross-Strait Relations plays only into a minority of people's most concerning issues when it comes to politics, as well as no strong discrepancies between administrative divisions across Taiwan. However, in **Map 4** we can see that there is a signifciant increase in importance between the pre- and post-period, especially in Lienchiang County (Northwest Islands) which are mere kilometers off the coast of China. While it cannot be said definitively that the Hong Kong protest caused this swing, it is quite striking how the islands closest to China so the most increase in salience of this issue.

Third, we will use **Map 5** and **Map 6** to display spatially the change and baseline levels in Taiwanese identity using the code below:

In [71]:
```
# set plot bounds
fig,ax = plt.subplots(ncols = 2, figsize = (20, 8))
ax1, ax2 = ax

# plot variable for map 5
geodf_join.plot(ax = ax1, column = "taiwanese_average", cmap = "Oranges", legend = True)
ax1.set_title("Map 5: Average Percent Identifying as Exclusively Taiwanese", fontsize = 12)
ax1.set_xlabel("Longitude", fontsize = 10)
ax1.set_ylabel("Latitude", fontsize = 10)

# plot variable for map 6
geodf_join.plot(ax = ax2, column = "both_change", cmap = "Blues_r", legend = True)
ax2.set_title("Map 6: Change in Percent Identifying as Both Taiwanese and Chinese", fontsize = 12)
ax2.set_xlabel("Longitude", fontsize = 10)
ax2.set_ylabel("Latitude", fontsize = 10)
```

Out[71]:
```
Text(1090.0697650950895, 0.5, 'Latitude')
```



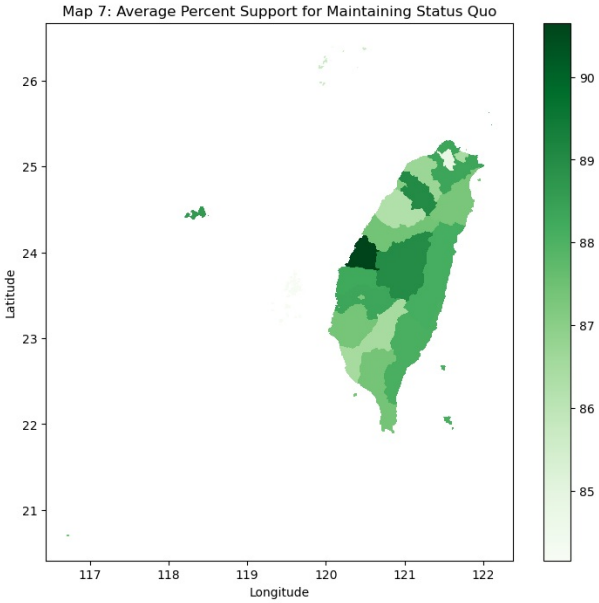


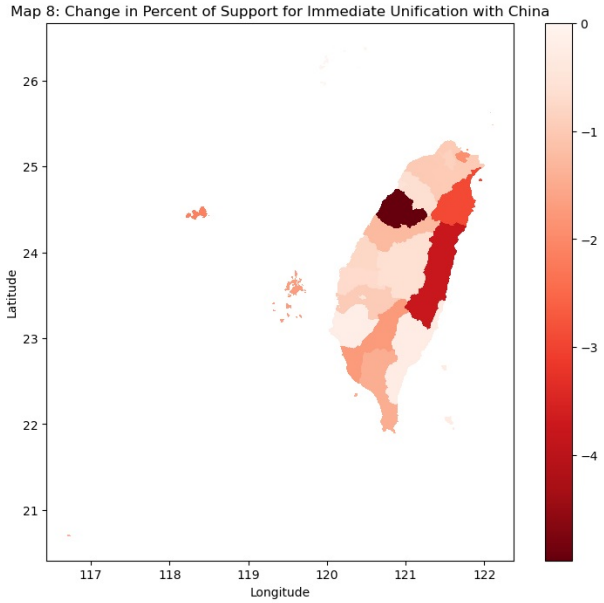
Looking at **Map 5** above, strong majorities or significantl minorities identify as exclusively Taiwanese across all surveys, and the value of identifying as at least partially Taiwanese also rises if we were to plot both **taiwanese** and **both** in **Map 5**. Likewise, this strong Taiwanese identity correlates with Tsai Ing-wen's presidential performance as the pro-independence party mentioned before, suggesting a correlation between identity and political party preference. Likewise, in **Map 6**, we can see that between the pre- and post-period there have been a significant collapse in those who identify as both Taiwanese and Chinese. Moreover, this collapse is strongest on the counties closest to China (the island counties). Furthermore, while not represented spatially above, the measuring of the change in identifying as **taiwanese** and **chinese** exclusively between pre- and post-period suggested that there was a growth in identifying as **taiwanese** across the board and a mixed increase and decrease in identifying as **chinese** across the board. This suggest that above all else, the Hong Kong protests correlate with a polarisation in identity in Taiwan with the collapse of people identifying as a dual-identity represented in **Map 6**.

And finally, for our fourth set of maps, **Map 7** and **Map 8** plot our **status** variable spatially showing the change in how people in Taiwan view the country's political status.

In [72]:
```
# set plot bounds
fig,ax = plt.subplots(ncols = 2, figsize = (20, 8))
ax1, ax2 = ax

# plot variable for map 7
geodf_join.plot(ax = ax1, column = "msq_average", cmap = "Greens", legend = True)
ax1.set_title("Map 7: Average Percent Support for Maintaining Status Quo", fontsize = 12)
ax1.set_xlabel("Longitude", fontsize = 10)
ax1.set_ylabel("Latitude", fontsize = 10)

# plot variable for map 8
geodf_join.plot(ax = ax2, column = "uni_change", cmap = "Reds_r", legend = True)
ax2.set_title("Map 8: Change in Percent of Support for Immediate Unification with China", fontsize = 12)
ax2.set_xlabel("Longitude", fontsize = 10)
ax2.set_ylabel("Latitude", fontsize = 10)
```

Out[72]:
```
Text(1090.0697650950895, 0.5, 'Latitude')
```





From **Map 7**, we can see that there is universal super majority support for maintaining the status quo with China across the board. Interestingly, this is lowest however in Taipei, where there is a larger percentage of "Mainlanders" (people who came to Taiwan in the last ~75 years) (BBC, 2022) and who dominated the political culture of Taiwan until the 90s. Furthermore, this trend does not seem to be reversing as between the pre- and post-period, support for immediate unification with China has declined margionally. While not plotted, we also found that increase support for independence *or* maintaining the status quo was not universal, but shifts towards either independence *and* maintaining the status quo was univeral as represented by the counterfactual in **Map 8**.

Overall from these 8 maps, we can see that there are strong changes in identity, status, and views on Cross-Strait Relations since the Hong Kong Protests. The general trend is a further increase in the strong Taiwanese identity and status, however it is impossible to say whether the protests themselves *caused* these shifts. Moreover, some of the limitations of the data include the surveys themselves. While some questions were not asked on some surveys, other questions like the respondent's gender showed a heavily skewed disribution towards **female**. Likewise, our analysis also shows that **taiwanese** and **chinese** are also political rather than solely ethnic identities. This is because the variables identifying parent's ancestry showed that there were respondents who were ethnically of chinese ancestry but identitifed exclusively as taiwanese and respondents who were ethnically taiwanese aborigional but identified exclusively as chinese. Furthermore, many respondents fell in the middle between taiwanese ancestry and chinese identity or visa versa. This suggests that the **identity** data is both useful as it is political in origin, but also perhaps not entirely representative of the poli-ethnic differences in Taiwan.

Part 2: Taipei Times Data

Wrangling Taipei Times Scraped Data

We begin this section by wrangling the data we scraped previously. The goal of wrangling the Taipei Times data is to find survey-level variables that can help explain the variation between the each TEDS survey. We first explore the data, cleaning and visualizing the type of language used and identifying common words per survey. Then, we extract the following variables:

- Sentiment. We calculate the sentiment per article and find the mean per survey, in addition to percentage positive, negative, and neutral.
- Article type. We extract the type of article, such as domestic politics, sports, business, etc.
- Topic modeling. We use Latent Dirichlet Allocation to find ten topics between all the articles. For example, one topic is related to the Hong Kong Protests. Finally, we create a survey-level data frame with the variables above, dummy variables for each survey, and interaction terms.

We read in the scraped data from a csv. The data was scraped previously in an external jupyter notebook file called **Web Scraping**. Additionally, we add the list of url's so that we can pull article type information for each article.

In [73]:
```
# read in data saved in Web_Scraping
df_china_results = pd.read_csv('Data/scraped_china_data.csv')
```

In [74]:
```
# combine with urls
```

```
df_china_urls = pd.read_csv('Data/china_link_list.csv')
df_china_results = pd.concat([df_china_results, df_china_urls], axis=1)
```

Sometimes, other links are scraped discussing unrelated topics. Particularly, if a website has a list of most read articles that are shown in a side window. Thus, we ensure that each article is discussing Mainland China.

In [75]:
```
# ensure all data is about China
df_china_results['China'] = 0
df_china_results.loc[df_china_results.text.str.contains('China'),'China'] = 1

# remove any not about China
df_china_results = df_china_results[df_china_results.China != 0]

# delete China column
df_china_results = df_china_results.drop(['China'], axis=1)
```

Next, we ensure that all dates are formatted so that we can do month and year analysis. Extra columns are deleted.

In [76]:
```
# ensure all data from correct dates (no 2023)
df_china_results[['Day', 'Month', 'Date', 'year']] = df_china_results.date.str.split(expand=True)

# delete columns with 2023
df_china_results = df_china_results[df_china_results.year != '2023']

# delete Day, Month, Date
df_china_results = df_china_results.drop(['Day'], axis=1)
df_china_results = df_china_results.drop(['Month'], axis=1)
df_china_results = df_china_results.drop(['Date'], axis=1)
```

We create a new column called 'survey' where we can identify each survey. The surveys are indexed based on data, with the first survey labeled 1. Since the first two surveys are distinct, but run on the same dates, they are labeled 1.1 and 1.2. Some of the surveys were not included in our final analysis, so any column not provided with an id and labeled 0 is removed.

In [77]:
```
# create column for survey no
df_china_results['survey'] = 0
df_china_results.loc[:292,'survey'] = 1.1 # survey 1.1
df_china_results.loc[298:590,'survey'] = 1.2 # survey 1.2
df_china_results.loc[596:880,'survey'] = 2 # survey 2
df_china_results.loc[886:1190,'survey'] = 3 # survey 3
df_china_results.loc[1196:1490,'survey'] = 4 # survey 4
df_china_results.loc[1497:1746,'survey'] = 5 # survey 5
df_china_results.loc[1752:1996,'survey'] = 6 # survey 6
df_china_results.loc[2002:2305,'survey'] = 7 # survey 7
df_china_results.loc[2311:2601,'survey'] = 8 # survey 8
df_china_results.loc[2607:2911,'survey'] = 9 # survey 9
df_china_results.loc[2917:3236,'survey'] = 10 # survey 10
df_china_results.loc[3242:3533,'survey'] = 11 # survey 11
df_china_results.loc[3539:3845,'survey'] = 12 # survey 12
df_china_results.loc[3851:4190,'survey'] = 13 # survey 13
df_china_results.loc[4196:4497,'survey'] = 14 # survey 14
df_china_results.loc[4503:4854,'survey'] = 15 # survey 15
df_china_results.loc[4861:5191,'survey'] = 16 # survey 16
df_china_results.loc[5198:5501,'survey'] = 17 # survey 18
df_china_results.loc[5507:5814,'survey'] = 18 # survey 17
df_china_results.loc[5820:6081,'survey'] = 19 # survey 19
```

The initial data frame is shown below:

In [78]:
```
df_china_results.head()
```

Out[78]:

	title	date	text	url	year	survey
0	BBC's Thai transmission ends as talks with jun...	Thu, Mar 09, 2017	BBC's Thai transmission ends as talks with ...	https://www.taipeitimes.com/News/world/archive...	2017	1.1
1	FEATURE: China 'comfort women' history buried ...	Thu, Mar 09, 2017	FEATURE: China 'comfort women' history buri...	https://www.taipeitimes.com/News/world/archive...	2017	1.1
2	Philippines aims for 'golden age of infrastruc...	Thu, Mar 09, 2017	Philippines aims for 'golden age of infrast...	https://www.taipeitimes.com/News/world/archive...	2017	1.1
3	China proposes US-N Korea suspension	Thu, Mar 09, 2017	China proposes US-N Korea suspension HEAD-...	https://www.taipeitimes.com/News/world/archive...	2017	1.1
4	Taiwan still attracting Chinese businesspeople	Thu, Mar 09, 2017	Taiwan still attracting Chinese businesspe...	https://www.taipeitimes.com/News/taiwan/archiv...	2017	1.1

The text in each of the articles is cleaned using the split and strip functions which make the data easier to work with.

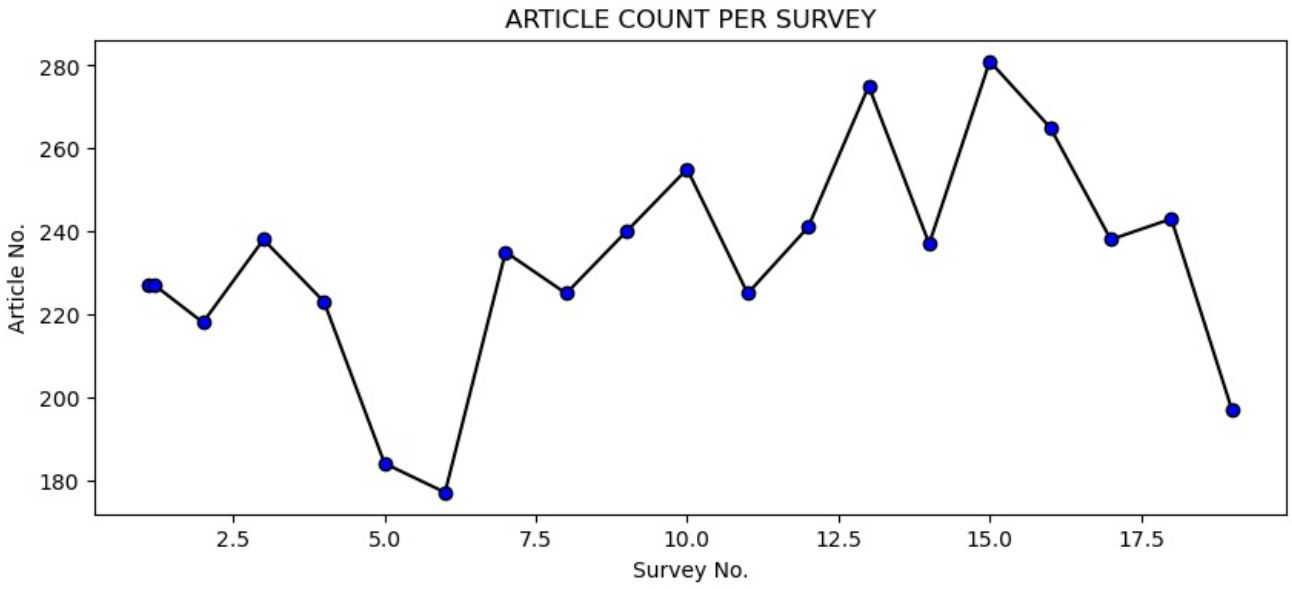
In [79]:
```
# seperate words for analysis
df_china_results['words'] = df_china_results.text.str.strip().str.split('[\W_]+')
```

Content Analysis

We begin our data visualization in this section through content analysis. This portion of the project is exploratory -- we are gaining a general understanding of the data and what to look for prior to determining appropriate survey-level variables. We include article count, most common word analysis, and visualize the article text in a word cloud.

The first graph shows article count per survey. Since for each survey data was scraped for the two weeks prior to the survey start, how many articles can provide important insight into whether there are several frequently reported events or limited information in the news cycle. The number of articles per survey range from about 170 to 280 articles, with a large dip for survey number 6 and more articles from survey 12 to 16.

In [80]:
```
# graph article count per survey
survey_count = df_china_results.groupby(['survey'])[['survey']].count()
survey_count.plot.line(color = 'black', marker = 'o', markerfacecolor = 'blue', legend=None, figsize=(10,4))
plt.xlabel("Survey No.")
plt.ylabel("Article No.")
plt.title("ARTICLE COUNT PER SURVEY");
```



To find commonly-used words we must remove stopwords which skew the results. For example, the word 'the' is used very often, but it does not tell us any important information. We used a custom stopwords list, which includes stopwords from CountWordsFree (CountWordsFree, no date) and additional topic-specific stopwords such as China, Taiwan, and news. Rather than using functions that remove stopwords automatically, we chose to import and remove stopwords from a custom list to better fit our project.

Our analysis is modelled off the pandas text analysis conducted by Jakub Nowacki, who explains in detail how to find word count per survey and plot the results (SigDelta, 2017).

We also clean the data to prepare for our analysis. We extract each word into a seperate row, ensure all words are lowercase, remove empty rows, and calculate counts per survey for each word.

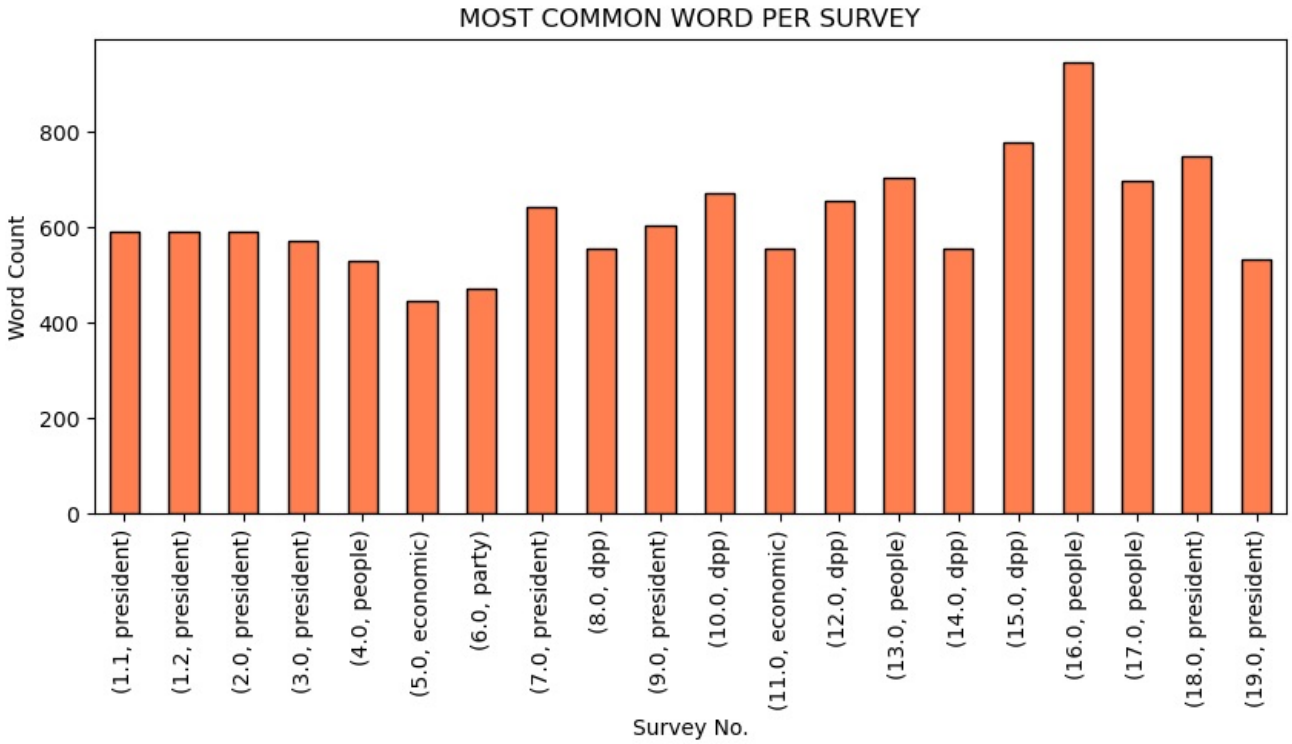
In [81]:
```
# import stopwords
stopwords = pd.read_csv('Data/stop_words_english.txt')
```

In [82]:
```
# most common words analysis

# break apart each word
rows = list()
for row in df_china_results[['survey', 'words']].iterrows():
    r = row[1]
    for word in r.words:
        rows.append((r.survey, word))

words = pd.DataFrame(rows, columns=['survey', 'word'])

# lowercase all words/ remove '01'
words['word'] = words.word.str.lower()
words['word'] = words.word.str.replace('01','')

# remove stopwords
words = words.assign(remove = words.word.isin(stopwords.stopwords).astype(int))

# remove rows with stop word
words = words[words.remove != 1]

# delete remove column
words = words.drop(['remove'], axis=1)

# remove empty strings
words = words[words.word.str.len() > 0]
```

```
# number of words per survey
counts = words.groupby('survey').word.value_counts().to_frame().rename(columns={'word':'n_w'})
```

The plot below shows the most common word per survey and its count. For a plurality of surveys, the most common word is 'president', likely referring to Taiwan's president Tsai or foreign leaders. The other most common words are people and economic, suggesting discussions of economic issues and policies. The other surveys include party and dpp, which is the Democratic Progressive Party, one of the two major political parties in Taiwan.

In [83]:
```
# plot most common word per survey
r = counts['n_w'].groupby(level=0).nlargest(1).reset_index(level=0, drop=True)
r.plot.bar(edgecolor = 'black', color = 'coral', figsize=(10,4))
plt.xlabel("Survey No.")
plt.ylabel("Word Count")
plt.title("MOST COMMON WORD PER SURVEY");
```



To visualize the results accross the text, we include a word cloud of the most common words and phrases in the *Taipei Times* articles. For example, 'green light' is likely referring to approving or authorizing something. 'Sijhih mrt' is the name for an mrt (mass rapid transport) rail line. Thus, 'mrt green' might refer to the authorization for the creation of the 'Sijhih mrt'. Other common phrases are related to exports, suggesting writing about trade and leadership, which likely discusses governance.

We modelled our word cloud off the word cloud analysis conducted by Aman Kharwal (The Clever Programmer, 2021).

In [84]:
```
# wordcloud
text = " ".join(i for i in words.word)
wordcloud = WordCloud(background_color="white", width=1800, height=800).generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title("TAIPEI TIMES: WORD CLOUD")
plt.show()
```

Sentiment Analysis

The section portion of our Taipei Times wrangling and visualization is sentiment analysis. We use TextBlob to calculate sentiment for each article. TextBlob uses a dictionary of words categorized as negative, positive, and neutral and provides an average sentiment

measurement per article. We also break apart the sentiment by positive, negative, and neutral (no negative articles were identified). We do this by classified negative as below -0.1, positive as greater than 0.1, and neutral in the middle.

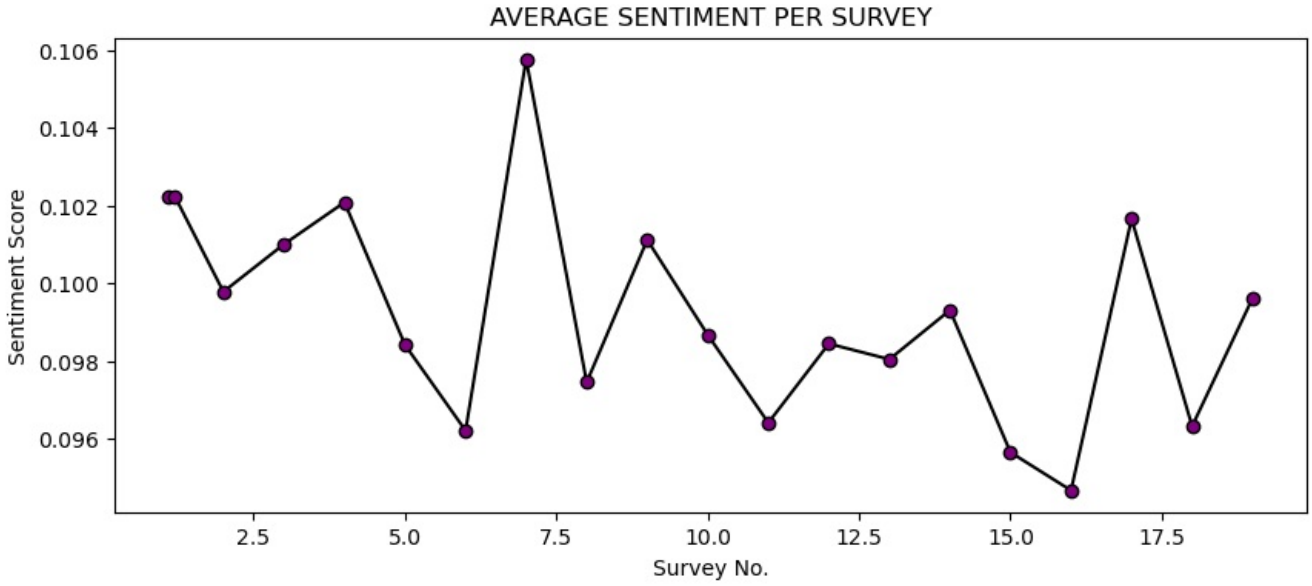
Our sentiment analysis is modelled off the sentiment analysis described by Natassha Selvaraj who details how to use TextBlob (Beyond Data Science, 2020).

In [85]:
```
# calculate text sentiment
df_china_results['text_sentiment'] = df_china_results['text'].apply(lambda abstract: TextBlob(abstract).sentime
```

In [86]:
```
# calculate negative, neutral, and positive sentiment
df_china_results['Labels'] = ['Negative' if x < -0.1 else 'Neutral' if -0.0999 < x < 0.1 else 'Positive' for x
```

The figure below shows the average sentiment per survey. Overall, most surveys appear fairly neutral, but there are differences between surveys. For example, survey 7 is quite positive, while surveys 10-16 appear more neutral. This figure shows how sentiment changes over time, but there seems to be fairly irregular trends contextual to the specific period of analysis rather than general shifts over time.

In [87]:
```
# graph average sentiment per survey
sentiment_mean = df_china_results.groupby(['survey'])[['text_sentiment']].mean()
sentiment_mean.plot.line(color = 'black', marker = 'o', markerfacecolor = 'purple', legend=None, figsize=(10,4)
plt.xlabel("Survey No.")
plt.ylabel("Sentiment Score")
plt.title("AVERAGE SENTIMENT PER SURVEY");
```



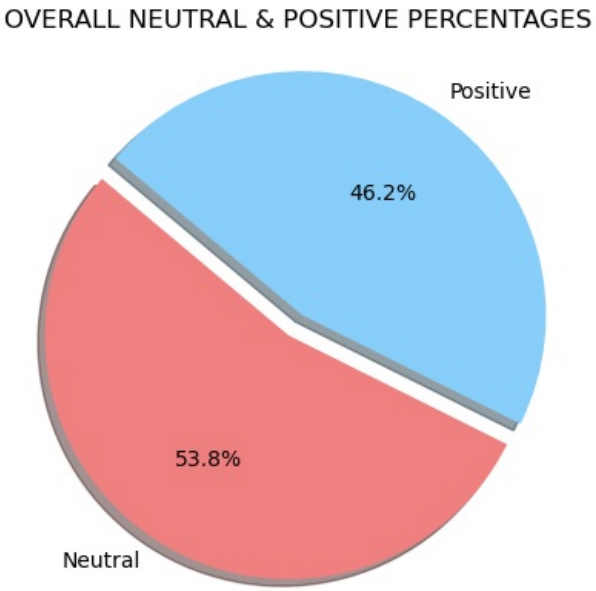
We decide to break down the data into positive and neutral sentiment (no articles are classified as negative) by creating a dummy variable for positive and neutral. We then calculate the percentage of articles per survey that are positive and neutral.

In [88]:
```
# dummy for neutral
df_china_results['Neutral'] = 0
df_china_results.loc[df_china_results.Labels.str.contains('Neutral'),'Neutral'] = 1

# dummy for positive
df_china_results['Positive'] = 0
df_china_results.loc[df_china_results.Labels.str.contains('Positive'),'Positive'] = 1

# count each per survey
sentiment_count = df_china_results.groupby(['survey'])[['Neutral', 'Positive']].sum()

# merge with article count for normalization
normalize_count = pd.concat([sentiment_count, survey_count], axis=1)

# calculate percentages
normalize_count['Neutral %'] = normalize_count['Neutral'] / normalize_count['survey'] * 100
normalize_count['Positive %'] = normalize_count['Positive'] / normalize_count['survey'] * 100

# remove extra columns
normalize_count = normalize_count.drop(['Neutral', 'Positive', 'survey'], axis=1)
```

The pie chart below shows the percentage accross surveys of positive and neutral sentiment. The majority of surveys (53.8%) are neutral and the remaining (46.2%) are positive. The Taipei Times appears to mostly provide neutral and mildly-positive analysis, suggesting fact-based rather than emotion-based reporting.

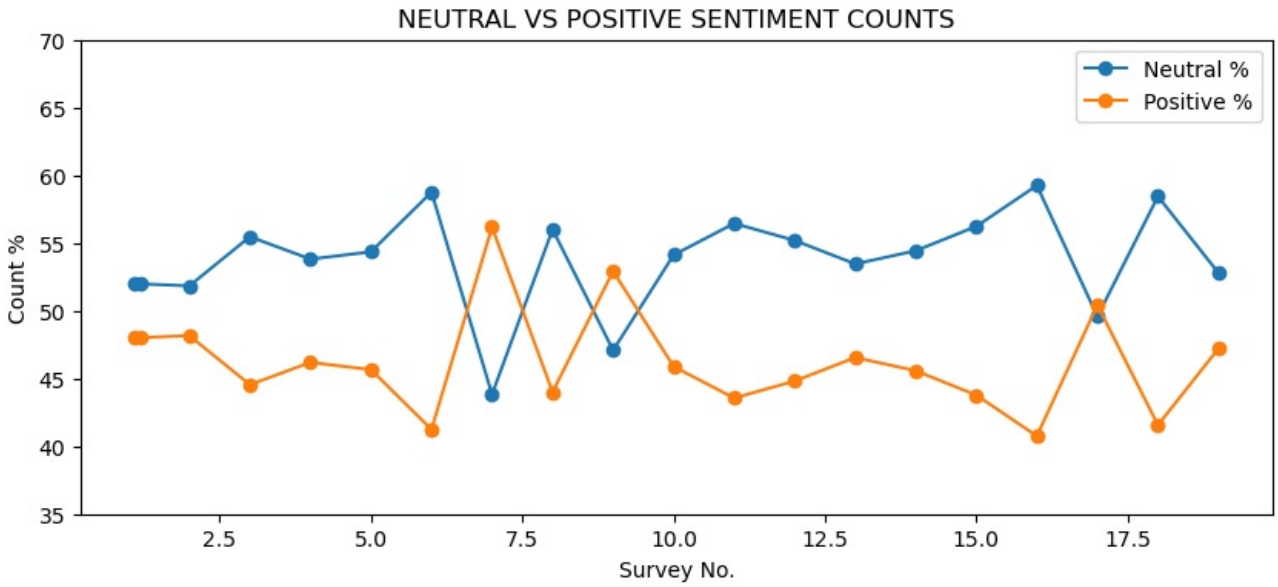
In [89]:
```
# pie chart

# count neutral/positive counts
neutral = df_china_results['Neutral'].sum()
positive = df_china_results['Positive'].sum()
```

```
# data
labels = 'Neutral', 'Positive'
sizes = [neutral, positive]
colors = ['lightcoral', 'lightskyblue']
explode = (0.1, 0)

# Plot
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)
plt.title("OVERALL NEUTRAL & POSITIVE PERCENTAGES")

plt.axis('equal')
plt.show()
```



The line chart below shows the percentage of positive and neutral sentiment for each survey. For the most part, there are significantly more neutral surveys, but this is not true for survey 7, 9, and 17.

In [90]:
```
# line chart
normalize_count.plot.line(figsize=(10,4), marker = 'o')
plt.xlabel("Survey No.")
plt.ylabel("Count %")
plt.ylim(35, 70)
plt.title("NEUTRAL VS POSITIVE SENTIMENT COUNTS");
```



Category Analysis

In this section, we investigate category analysis. The *Taipei Times* breaks articles into the following categories: world, Taiwan, sport, front page, editorials, business, and featured articles. The article type is identified in the url so we extract it and create dummy variables for article type. Then, we calculate the percentage of each type of article per survey.

In [91]:
```
# identify percentage of each article type

# world
```

```
df_china_results['world'] = 0
df_china_results.loc[df_china_results.url.str.contains('world'),'world'] = 1

# Taiwan
df_china_results['taiwan'] = 0
df_china_results.loc[df_china_results.url.str.contains('taiwan'),'taiwan'] = 1

# Sports
df_china_results['sport'] = 0
df_china_results.loc[df_china_results.url.str.contains('sport'),'sport'] = 1

# Front
df_china_results['front'] = 0
df_china_results.loc[df_china_results.url.str.contains('front'),'front'] = 1

# Editorials
df_china_results['editorial'] = 0
df_china_results.loc[df_china_results.url.str.contains('editorial'),'editorial'] = 1

# Biz
df_china_results['biz'] = 0
df_china_results.loc[df_china_results.url.str.contains('biz'),'biz'] = 1

# Feat
df_china_results['feat'] = 0
df_china_results.loc[df_china_results.url.str.contains('feat'),'feat'] = 1

# article type distribution
article_type = df_china_results.groupby(['survey'])[['world', 'taiwan', 'sport', 'front', 'editorial', 'biz', '

# merge with article count for normalization
normalize_type = pd.concat([article_type, survey_count], axis=1)

# calculate percentages
normalize_type['world %'] = normalize_type['world'] / normalize_type['survey'] * 100
normalize_type['taiwan %'] = normalize_type['taiwan'] / normalize_type['survey'] * 100
normalize_type['sports %'] = normalize_type['sport'] / normalize_type['survey'] * 100
normalize_type['front %'] = normalize_type['front'] / normalize_type['survey'] * 100
normalize_type['editorial %'] = normalize_type['editorial'] / normalize_type['survey'] * 100
normalize_type['business %'] = normalize_type['biz'] / normalize_type['survey'] * 100
normalize_type['featured %'] = normalize_type['feat'] / normalize_type['survey'] * 100

# remove extra columns
normalize_type = normalize_type.drop(['world', 'taiwan', 'sport', 'front', 'editorial', 'biz', 'feat', 'survey'
```

The normalize_type data frame shows the percentage of each article per survey. For example, 'sports' and 'featured' are relatively rare, while 'taiwan' and 'business' are common article types.

In [92]:
```
normalize_type
```

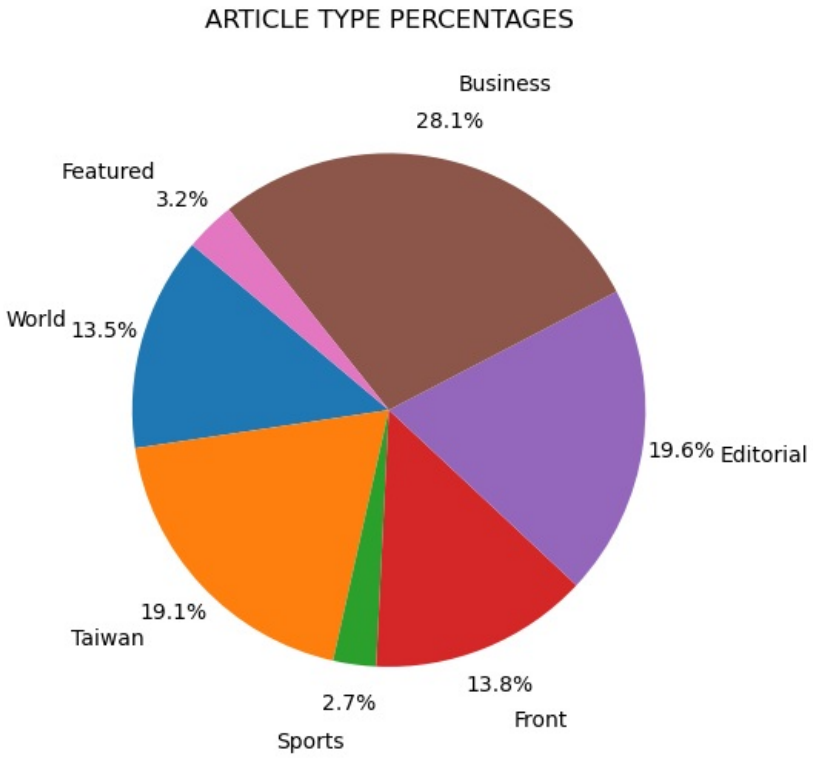
Out[92]:

survey	world %	taiwan %	sports %	front %	editorial %	business %	featured %
1.1	12.334802	22.907489	3.083700	13.656388	18.502203	25.110132	3.964758
1.2	12.334802	22.907489	3.083700	13.656388	18.502203	25.110132	3.964758
2.0	15.596330	22.018349	0.917431	11.009174	18.807339	28.899083	2.293578
3.0	15.966387	18.487395	3.781513	10.504202	17.226891	30.252101	2.941176
4.0	9.865471	19.730942	3.587444	13.004484	18.385650	31.390135	3.139013
5.0	8.152174	17.934783	2.717391	11.413043	20.108696	35.869565	3.804348
6.0	15.254237	15.819209	1.129944	14.124294	18.644068	29.943503	5.084746
7.0	11.489362	22.553191	2.127660	15.744681	21.276596	23.404255	2.553191
8.0	10.666667	17.777778	3.111111	15.555556	17.777778	32.000000	2.666667
9.0	13.333333	22.083333	2.500000	14.583333	18.333333	25.416667	2.916667
10.0	14.509804	16.470588	4.705882	16.078431	19.215686	26.274510	2.352941
11.0	14.222222	16.000000	1.777778	8.000000	20.000000	35.555556	3.555556
12.0	9.128631	18.257261	4.149378	15.767635	19.502075	29.045643	2.904564
13.0	10.545455	20.727273	2.181818	14.181818	19.636364	28.000000	4.363636
14.0	9.704641	18.987342	4.219409	12.658228	21.097046	29.535865	3.375527
15.0	11.743772	19.217082	1.067616	14.234875	18.861210	30.960854	3.914591
16.0	20.754717	18.490566	3.396226	11.320755	20.754717	22.264151	2.264151
17.0	15.126050	17.647059	2.521008	15.126050	21.428571	26.050420	2.100840
18.0	21.399177	13.580247	1.646091	18.106996	22.633745	19.753086	2.880658
19.0	15.228426	18.274112	1.522843	15.736041	18.274112	27.411168	2.030457

The pie chart below shows the overall percentage of each article type. Business comprises 28.1% of articles, editorials 19.6%, front page

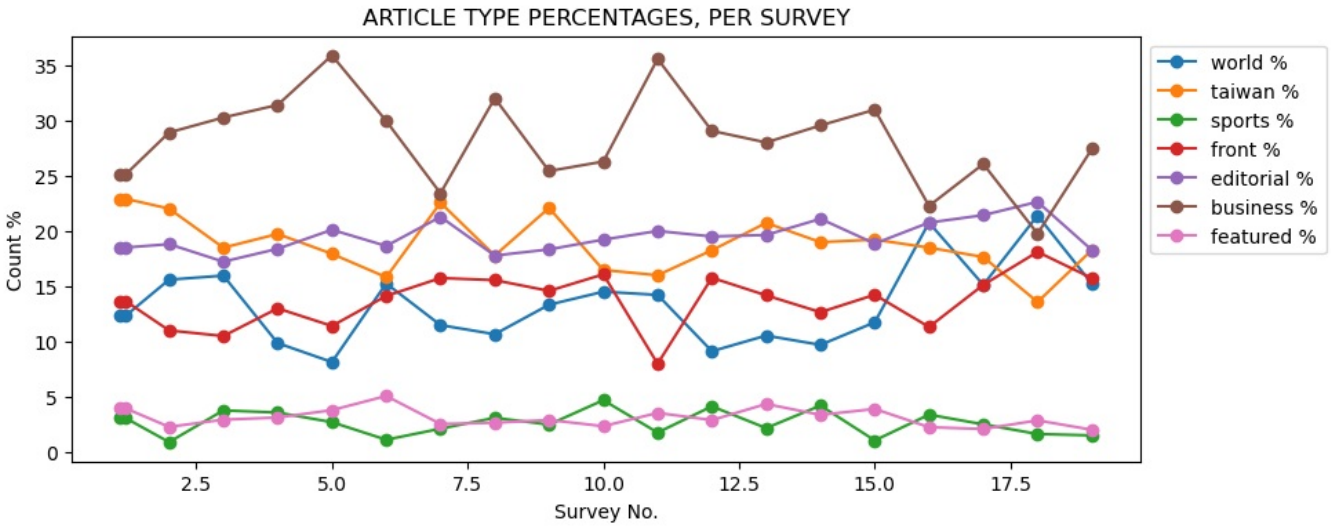
13.8%, sports 2.7%, Taiwan 19.1%, world 13.5%, and featured 3.2%. However, this chart just shows overall averages, without a per-survey breakdown.

In [93]:
```
# pie chart

# count each category
world = df_china_results['world'].sum()
taiwan = df_china_results['taiwan'].sum()
sport = df_china_results['sport'].sum()
front = df_china_results['front'].sum()
editorials = df_china_results['editorial'].sum()
biz = df_china_results['biz'].sum()
feat = df_china_results['feat'].sum()

# data
labels = 'World', 'Taiwan', 'Sports', 'Front', 'Editorial', 'Business', 'Featured'
sizes = [world, taiwan, sport, front, editorials, biz, feat]

# plot
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140, pctdistance=1.15, labeldistance = 1.3)
plt.title("ARTICLE TYPE PERCENTAGES", x=0.5, y=1.15)

plt.axis('equal')
plt.show()
```



The line chart below shows the percentage of each article type, per survey. Over time, the category percentages remain relatively stable. Sports and featured articles are the least common and business is typically the most common type.

In [94]:
```
# line chart
normalize_type.plot.line(figsize=(10,4), marker = 'o')
plt.xlabel("Survey No.")
plt.ylabel("Count %")
plt.legend(bbox_to_anchor=(1,1), loc="upper left")
plt.title("ARTICLE TYPE PERCENTAGES, PER SURVEY");
```



Topic Modeling

Finally, we use Latent Dirichlet Allocation to model ten topics accross the articles. We choose a **max_df** of 50%, meaning that words can only be included in max 50% of articles. This number is normally higher, but given the massive amount of data we used and that we want to find common topics, we set the percentage lower. We choose a **min_df** of 2, which means that a word must be included twice, which removes really niche words. We remove **stop_words** seperately in this model because we want to include topic-specific words like China which were removed before.

When running the model, we set the sead **random_state** as 1, so when the project is rerun the same results are generated. We choose ten topics because there is a significant amount of data, so we generate a substantial number of topics.

We base our topic modeling analysis off the code described by Shashank Kapadia who details how to modify a Latent Dirichlet Allocation model and how to present the results (Beyond Data Science, 2019).

In [95]:
```
# create document term matrix for topic modeling
count_vect = CountVectorizer(max_df=0.5, min_df=2, stop_words='english')
doc_term_matrix = count_vect.fit_transform(df_china_results['text'].values.astype('U'))
```

In [96]:
```
# use Latent Dirichlet Allocation to find ten topics
LDA = LatentDirichletAllocation(n_components=10, random_state=1)
LDA.fit(doc_term_matrix)
```

Out[96]:
```
LatentDirichletAllocation(random_state=1)
```

We then print out the most common words per topic. The words clearly appear to discuss a specific topic. For example, topic #1 is likely referring to the Hong Kong protests.

In [97]:
```
# print top ten words for each topic
for i,topic in enumerate(LDA.components_):
    print(f'Top 10 words for topic #{i}:')
    print([count_vect.get_feature_names_out()[i] for i in topic.argsort()[-10:]])
    print('\n')
```

```
Top 10 words for topic #0:
['use', 'europe', 'waste', 'internet', 'companies', 'company', 'coal', 'energy', 'money', 'eu']


Top 10 words for topic #1:
['protesters', 'beijing', 'pro', 'protests', 'democracy', 'police', 'law', 'territory', 'kong', 'hong']


Top 10 words for topic #2:
['chen', 'barclays', 'dual', 'track', 'william', 'reservist', 'training', 'vice', 'reservists', 'program']


Top 10 words for topic #3:
['rabbit', 'kishida', '72', 'nguyen', 'plane', 'vietnamese', 'japanese', 'medvedev', 'cat', 'vietnam']


Top 10 words for topic #4:
['company', 'lighthouse', 'beri', 'utilization', 'plant', 'report', 'production', 'ase', 'investment', 'technol
ogy']


Top 10 words for topic #5:
['beri', 'lighthouse', 'utilization', 'plant', 'quarter', 'report', 'production', 'ase', 'technology', 'investm
ent']


Top 10 words for topic #6:
['life', 'tsai', 'center', 'political', 'japanese', 'school', 'university', 'history', 'kmt', 'india']


Top 10 words for topic #7:
['city', 'landmarks', 'maps', 'cdc', 'cultural', 'disease', 'google', 'reported', 'diarrhea', 'cases']


Top 10 words for topic #8:
['market', 'institute', 'region', 'pacific', 'nhia', 'dollars', 'countries', 'india', 'chips', 'beijing']


Top 10 words for topic #9:
['league', 'going', 'open', 'attiyah', 'old', 'signing', 'al', 'match', 'lee', 'second']
```

We then append the results to the original dataframe and create and rename dummy variables for each topic.

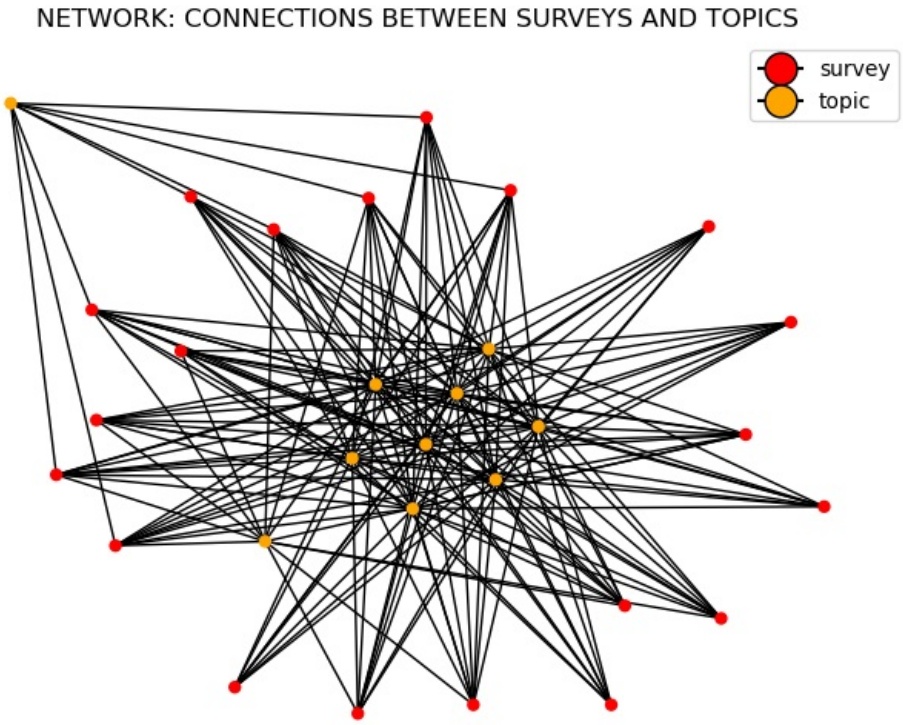
In [98]:
```
# clean/modify dataframe

# append topics to dataframe
topic_values = LDA.transform(doc_term_matrix)
topic_values.shape
df_china_results['topic'] = topic_values.argmax(axis=1)

# add dummy variables
dummies = pd.get_dummies(df_china_results['topic'])
df_china_results = pd.concat([df_china_results, dummies], axis=1)

# rename columns
df_china_results = df_china_results.rename(columns={0: 'topic_0', 1: 'topic_1', 2: 'topic_2', 3: 'topic_3', 4:
                                                    5: 'topic_5', 6: 'topic_6', 7: 'topic_7', 8: 'topic_8', 9:
```

We conduct network analysis which allows us to explore the relationship between topics and each survey. While the network does not show any detailed patterns, the results are clear that there is substantial connection between each survey and topic. Thus, the topics are quite engrained in each period rather than contextual to each topic.

The network analysis package we used does not automatically add a legend. To create a legend, we adopted the method explained by Baarath Srinivasan (Stack Overflow, 2022).

In [99]:
```
# network analysis

# vertex
vertex = df_china_results[['survey']].copy()
vertex = vertex.drop_duplicates(keep='first')
vertex['survey_no'] = 'survey-' + vertex['survey'].astype(str)
vertex = vertex.drop(['survey'], axis=1)

# edges
edges = df_china_results[['survey', 'topic']].copy()
edges['survey_no'] = 'survey-' + edges['survey'].astype(str)
edges['topic_no'] = 'topic-' + edges['topic'].astype(str)
edges = edges.drop(['survey', 'topic'], axis=1)

# create graph
G = nx.Graph()

# nodes
nodes_attr = vertex.set_index('survey_no').to_dict(orient = 'index')
```

```
nx.set_node_attributes(G, nodes_attr)

# edges
G = nx.from_pandas_edgelist(edges, source = 'survey_no', target = 'topic_no')

# legend
legend_elements = [
    Line2D([0], [0], marker='o', color='black', label='survey', markerfacecolor='red', markersize=15),
    Line2D([0], [0], marker='o', color='black', label='topic', markerfacecolor='orange', markersize=15)]

# custom colors
color = []
for node in G:
    if 'survey' in node:
        color.append('red')
    else:
        color.append('orange')

# results
nx.draw_spring(G, node_color=color, node_size=25)
plt.legend(handles=legend_elements, loc='upper right')
plt.title('NETWORK: CONNECTIONS BETWEEN SURVEYS AND TOPICS');
```



We calculate the percentage of each topic per data frame and remove extra columns.

In [100…

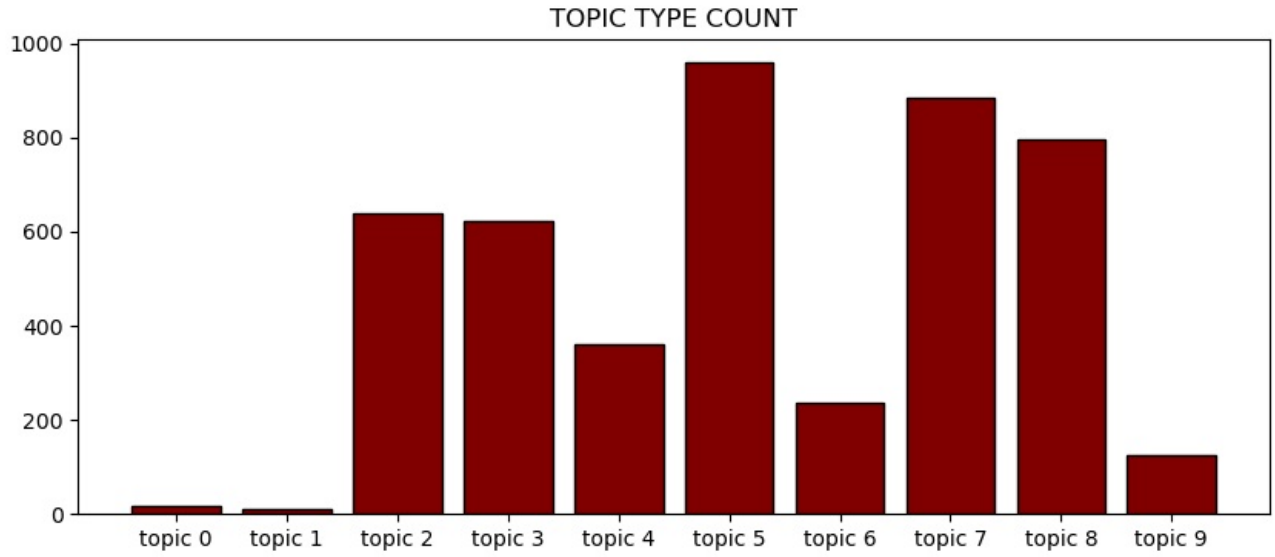
```
# calculate topic percentages

# groupby survey for each topic
topic_results = df_china_results.groupby(['survey'])[['topic_0', 'topic_1', 'topic_2', 'topic_3', 'topic_4', 't
                                                      'topic_6', 'topic_7', 'topic_8', 'topic_9']].sum()

# calculate percentages
topic_results['topic 0'] = topic_results['topic_0'] / survey_count['survey'] * 100
topic_results['topic 1'] = topic_results['topic_1'] / survey_count['survey'] * 100
topic_results['topic 2'] = topic_results['topic_2'] / survey_count['survey'] * 100
topic_results['topic 3'] = topic_results['topic_3'] / survey_count['survey'] * 100
topic_results['topic 4'] = topic_results['topic_4'] / survey_count['survey'] * 100
topic_results['topic 5'] = topic_results['topic_5'] / survey_count['survey'] * 100
topic_results['topic 6'] = topic_results['topic_6'] / survey_count['survey'] * 100
topic_results['topic 7'] = topic_results['topic_7'] / survey_count['survey'] * 100
topic_results['topic 8'] = topic_results['topic_8'] / survey_count['survey'] * 100
topic_results['topic 9'] = topic_results['topic_9'] / survey_count['survey'] * 100

# remove extra columns
topic_results = topic_results.drop(['topic_0', 'topic_1', 'topic_2', 'topic_3', 'topic_4', 'topic_5', 'topic_6'
                                    'topic_8', 'topic_9'], axis=1)
```

The bar chart below shows the count of each topic. Most of the topics are relatively common, except topics 0, 1, and 9.

In [101…

```
# pie chart

# count each category
topic_0 = df_china_results['topic_0'].sum()
topic_1 = df_china_results['topic_1'].sum()
```

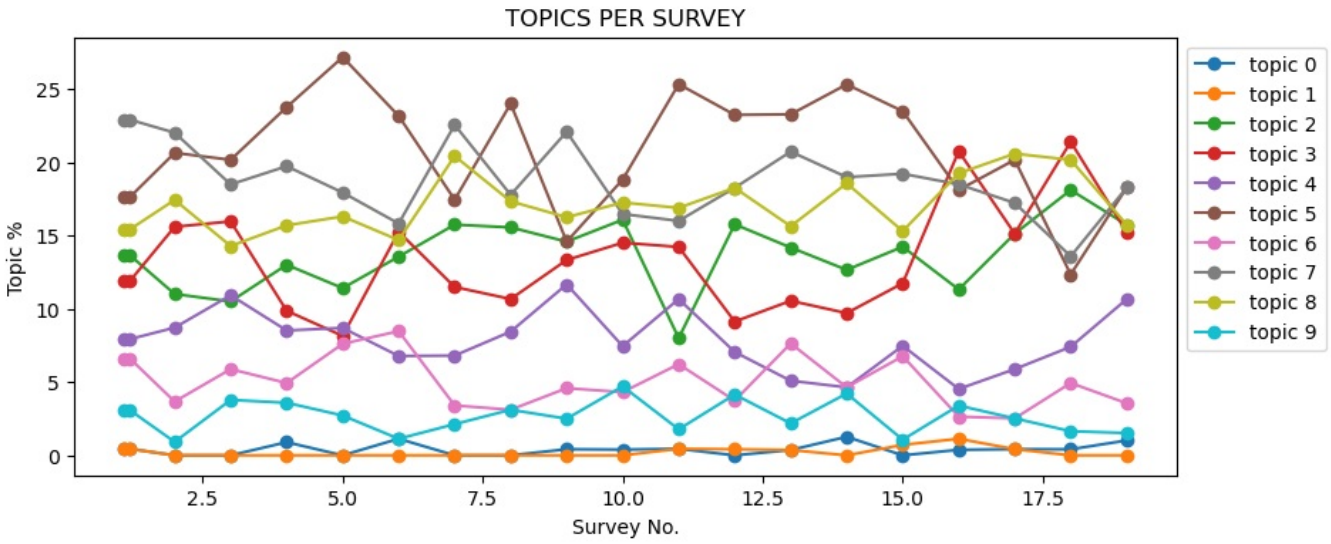
```
topic_2 = df_china_results['topic_2'].sum()
topic_3 = df_china_results['topic_3'].sum()
topic_4 = df_china_results['topic_4'].sum()
topic_5 = df_china_results['topic_5'].sum()
topic_6 = df_china_results['topic_6'].sum()
topic_7 = df_china_results['topic_7'].sum()
topic_8 = df_china_results['topic_8'].sum()
topic_9 = df_china_results['topic_9'].sum()

# data
labels = 'topic 0', 'topic 1', 'topic 2', 'topic 3', 'topic 4', 'topic 5', 'topic 6', 'topic 7', 'topic 8', 'to
sizes = [topic_0, topic_1, topic_2, topic_3, topic_4, topic_5, topic_6, topic_7, topic_8, topic_9]

# Plot
fig = plt.figure(figsize = (10, 4))
plt.bar(labels, sizes, color ='maroon', edgecolor = 'black')
plt.title("TOPIC TYPE COUNT");
```



The line chart below shows the distribution of topics per survey. The results often change depending on the survey, suggesting that topics are often modeling salient events which could provide an important proxy for external variation between surveys.

In [102…

```
# line chart
topic_results.plot.line(figsize=(10,4), marker = 'o')
plt.xlabel("Survey No.")
plt.ylabel("Topic %")
plt.legend(bbox_to_anchor=(1,1), loc="upper left")
plt.title("TOPICS PER SURVEY");
```



Preparing Data for Survey Data Comparisons

Finally, we aggregate the dataframes into a final dataframe designed to measure external survey variation. We concat the article counts, sentiment information, article types, and topic results information. Then, we create dummy variables for each survey so we can text the impact of each period leading up to the surveys.

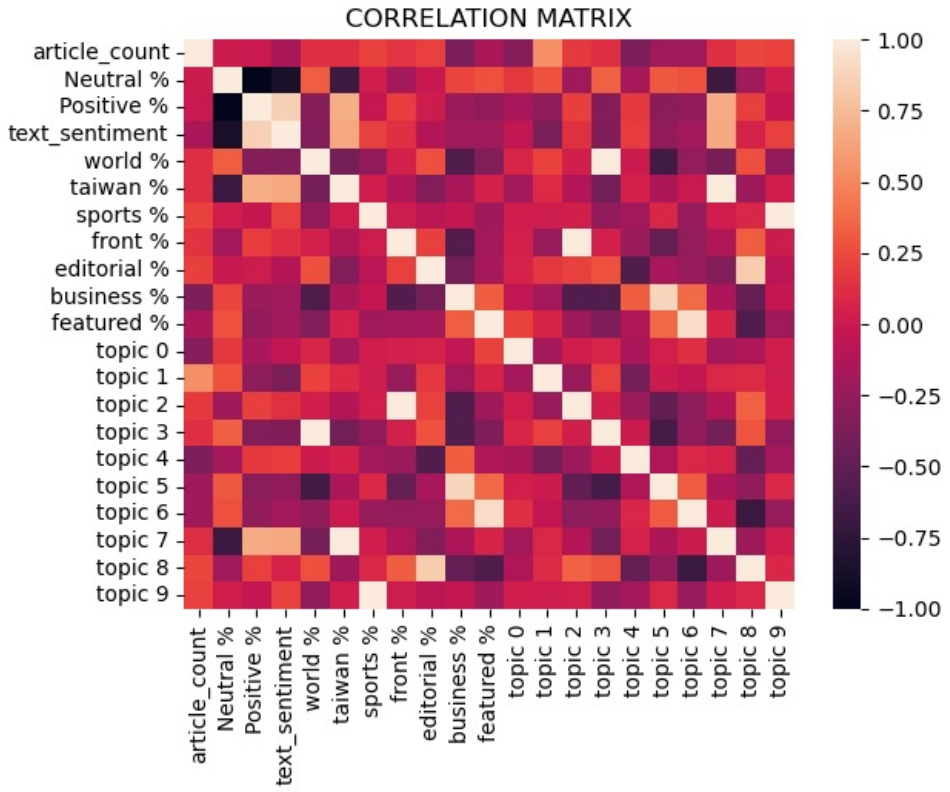
In [103…

```
# combine important variables for testing
external_variation_df = pd.concat([normalize_count, sentiment_mean, normalize_type, survey_count, topic_results
external_variation_df = external_variation_df.rename(columns = {'survey':'article_count'})
```

```
# create dummies
external_variation_df.reset_index(inplace=True)
dummies = pd.get_dummies(external_variation_df['survey'])
external_variation_df = external_variation_df.join(dummies)
```

In the correlation matrix below, we show the correlative relationships between each variable in the survey-level data frame (excluding interaction terms). Lighter colours are highly, positively correlated and darker colors are highly-negatively correlated. Some of the results are not particularly explanatory, for example that neutral and positive % are highly-negatively correlated. However, the matrix also shows some very interesting results. For example, topics 2 and 3 are often business-related and topic 7 is quite neutral.

In [104…
```
# Correlation Matrix
df_correl = external_variation_df[['article_count', 'Neutral %', 'Positive %', 'text_sentiment', 'world %', 'ta
                                   'sports %', 'front %', 'editorial %', 'business %', 'featured %', 'topic 0',
                                   'topic 2', 'topic 3', 'topic 4', 'topic 5', 'topic 6', 'topic 7', 'topic 8',
                                   'topic 9']].copy()
corr_matrix = df_correl.corr()
sns.heatmap(corr_matrix, annot=False)
plt.title("CORRELATION MATRIX")
plt.show()
```



The external_variation_df shows the final survey-level data frame, including number of articles, sentiment scores, article type, and article topics.

In [105…
```
# final per-survey dataframe
external_variation_df.head()
```

Out[105]:

	survey	Neutral %	Positive %	text_sentiment	world %	taiwan %	sports %	front %	editorial %	business %	...	10.0	11.0	12.0	13.0
0	1.1	51.982379	48.017621	0.102227	12.334802	22.907489	3.083700	13.656388	18.502203	25.110132	...	0	0	0	0
1	1.2	51.982379	48.017621	0.102227	12.334802	22.907489	3.083700	13.656388	18.502203	25.110132	...	0	0	0	0
2	2.0	51.834862	48.165138	0.099779	15.596330	22.018349	0.917431	11.009174	18.807339	28.899083	...	0	0	0	0
3	3.0	55.462185	44.537815	0.101002	15.966387	18.487395	3.781513	10.504202	17.226891	30.252101	...	0	0	0	0
4	4.0	53.811659	46.188341	0.102081	9.865471	19.730942	3.587444	13.004484	18.385650	31.390135	...	0	0	0	0

5 rows × 42 columns

Finally, we save the survey level data into a csv so it can be easily pulled up in future code if needed.

In [106…
```
# save as csv
survey_level_data = external_variation_df.to_csv('Data/survey_level_data.csv')
```

Part 3: Taipei Times & Public Opinion Data Analysis

In this section, we

In this section, we

1. Aggregate the data by survey and visualize variable relationships,
2. Conduct panel data analysis running regressions exploring the relationship between variables identifying Taiwanese perceptions of Mainland China and the variables derived from the scraped data.

Wrangle datasets to aggregate data

Combine Data

We then combine the aggregated survey data with the aggregated scraped data variables, creating one dataset for each dependent variable.

In [107…
```
# Tsai CSR dataset
tsai_csr_all = pd.merge(tsai_csr_df, external_variation_df, left_index=True, right_on='survey')

# CSR key dataset
csr_important_all = pd.merge(most_impot_df, external_variation_df, left_index=True, right_on='survey')

# identity dataset
identity_all = pd.merge(identity_df, external_variation_df, left_index=True, right_on='survey')

# status dataset
status_all = pd.merge(status_df, external_variation_df, left_index=True, right_on='survey')
```

Each dataset is visualized below.

In [108…
```
tsai_csr_all
```

Out[108]:

	survey	survey_x	tsai_csr nan	tsai_csr not satisfied at all	tsai_csr not specified	tsai_csr not very satisfied	tsai_csr satisfied	tsai_csr very satisfied	survey_y	Neutral %	...	10.0	11.0	12.0	13.0	14.0
0	1.1	100.0	0.0	34.695731	0.000000	32.152589	27.702089	5.449591	1.1	51.982379	...	0	0	0	0	0
1	1.2	100.0	0.0	22.906404	19.129721	30.706076	22.249589	5.008210	1.2	51.982379	...	0	0	0	0	0
3	3.0	100.0	0.0	29.837398	17.317073	30.813008	18.617886	3.414634	3.0	55.462185	...	0	0	0	0	0
4	4.0	100.0	0.0	24.776604	16.978067	32.900081	20.877335	4.467912	4.0	53.811659	...	0	0	0	0	0
5	5.0	100.0	0.0	28.706625	18.059937	30.520505	20.268139	2.444795	5.0	54.347826	...	0	0	0	0	0
7	7.0	100.0	0.0	28.767123	18.291700	31.990330	16.680097	4.270749	7.0	43.829787	...	0	0	0	0	0
9	9.0	100.0	0.0	34.843493	15.238880	30.313015	15.074135	4.530478	9.0	47.083333	...	0	0	0	0	0
10	10.0	100.0	0.0	35.250000	15.333333	29.250000	15.666667	4.500000	10.0	54.117647	...	1	0	0	0	0
11	11.0	100.0	0.0	34.372436	15.258409	33.552092	13.289582	3.527482	11.0	56.444444	...	0	1	0	0	0
13	13.0	100.0	0.0	22.637363	11.648352	22.564103	26.007326	17.142857	13.0	53.454545	...	0	0	0	1	0
14	14.0	100.0	0.0	28.790323	10.161290	21.290323	25.483871	14.274194	14.0	54.430380	...	0	0	0	0	1
15	15.0	100.0	0.0	19.920635	11.746032	20.000000	29.841270	18.492063	15.0	56.227758	...	0	0	0	0	0
16	16.0	100.0	0.0	12.009238	10.161663	17.859892	36.181678	23.787529	16.0	59.245283	...	0	0	0	0	0
17	17.0	100.0	0.0	17.091208	9.695974	17.995070	36.647494	18.570255	17.0	49.579832	...	0	0	0	0	0
18	18.0	100.0	0.0	17.957166	10.543657	22.075783	30.065898	19.357496	18.0	58.436214	...	0	0	0	0	0
19	19.0	100.0	0.0	21.770335	9.968102	19.457735	33.173844	15.629984	19.0	52.791878	...	0	0	0	0	0

16 rows × 50 columns

In [109…
```
csr_important_all
```

Out[109]:

	survey	survey_x	most_import aboriginal issues	most_import agricultural policy	most_import animal protection	most_import annuity reform	most_import built environment	most_import cabinet reshuffle	most_import chen, shui-bian amnesty issue	most_import competition between pan-blue and the pan-green	...	10.0
0	1.1	100.0	0.000000	0.000000	0.000000	18.891916	0.000000	0.000000	0.000000	0.000000	...	0
1	1.2	100.0	0.000000	0.000000	0.000000	14.203612	0.000000	0.000000	0.000000	0.000000	...	0
3	3.0	100.0	0.000000	0.000000	0.000000	13.902439	0.000000	0.000000	0.000000	0.000000	...	0
4	4.0	100.0	0.000000	0.000000	0.000000	9.504468	0.000000	0.000000	0.081235	0.081235	...	0
5	5.0	100.0	0.000000	0.078864	0.078864	8.280757	0.000000	0.000000	0.000000	0.000000	...	0
7	7.0	100.0	0.000000	0.000000	0.000000	7.896857	0.080580	0.000000	0.000000	0.000000	...	0
8	8.0	100.0	0.000000	0.000000	0.000000	11.924119	0.000000	0.000000	0.000000	0.000000	...	0
9	9.0	100.0	0.000000	0.164745	0.000000	6.342669	0.000000	0.000000	0.082372	0.000000	...	0
10	10.0	100.0	0.083333	0.000000	0.000000	4.666667	0.000000	0.000000	0.000000	0.000000	...	1
11	11.0	100.0	0.000000	0.082034	0.000000	5.414274	0.000000	0.000000	0.000000	0.000000	...	0
13	13.0	100.0	0.000000	0.000000	0.000000	6.666667	0.000000	0.000000	0.000000	0.073260	...	0
14	14.0	100.0	0.000000	0.080645	0.000000	5.887097	0.000000	0.000000	0.000000	0.000000	...	0
15	15.0	100.0	0.000000	0.000000	0.000000	7.460317	0.158730	0.000000	0.000000	0.000000	...	0
16	16.0	100.0	0.000000	0.000000	0.000000	6.928406	0.076982	0.000000	0.000000	0.000000	...	0
17	17.0	100.0	0.000000	0.000000	0.000000	6.409203	0.000000	0.000000	0.000000	0.000000	...	0
18	18.0	100.0	0.000000	0.000000	0.000000	7.166392	0.082372	0.082372	0.000000	0.000000	...	0
19	19.0	100.0	0.000000	0.079745	0.000000	7.496013	0.000000	0.000000	0.000000	0.000000	...	0

17 rows × 85 columns

In [110…
```
identity_all
```

Out[110]:

	survey	survey_x	identity both	identity chinese	identity it depends	identity nan	identity not specified	identity taiwanese	survey_y	Neutral %	...	10.0	11.0	12.0	13.0	14.0	1
0	1.1	100.0	2.179837	44.504995	0.000000	0.0	0.000000	53.315168	1.1	51.982379	...	0	0	0	0	0	
1	1.2	100.0	41.543514	2.791461	0.656814	0.0	2.545156	52.463054	1.2	51.982379	...	0	0	0	0	0	
2	2.0	100.0	47.714048	2.327515	0.000000	0.0	0.000000	49.958437	2.0	51.834862	...	0	0	0	0	0	
3	3.0	100.0	39.024390	3.658537	0.325203	0.0	2.764228	54.227642	3.0	55.462185	...	0	0	0	0	0	
4	4.0	100.0	38.505280	3.005686	0.487409	0.0	2.924452	55.077173	4.0	53.811659	...	0	0	0	0	0	
5	5.0	100.0	38.328076	3.470032	0.473186	0.0	3.627760	54.100946	5.0	54.347826	...	0	0	0	0	0	
6	6.0	100.0	46.626984	3.075397	0.000000	0.0	0.000000	50.297619	6.0	58.757062	...	0	0	0	0	0	
7	7.0	100.0	38.114424	3.545528	0.080580	0.0	2.175665	56.083803	7.0	43.829787	...	0	0	0	0	0	
8	8.0	100.0	45.257453	2.439024	0.000000	0.0	1.626016	50.677507	8.0	56.000000	...	0	0	0	0	0	
9	9.0	100.0	40.609555	4.200988	0.082372	0.0	3.130148	51.976936	9.0	47.083333	...	0	0	0	0	0	
10	10.0	100.0	43.666667	3.916667	0.333333	0.0	3.000000	49.083333	10.0	54.117647	...	1	0	0	0	0	
11	11.0	100.0	42.575882	3.035275	0.082034	0.0	3.773585	50.533224	11.0	56.444444	...	0	1	0	0	0	
13	13.0	100.0	35.164835	2.857143	0.659341	0.0	2.490842	58.827839	13.0	53.454545	...	0	0	0	1	0	
14	14.0	100.0	38.467742	3.790323	0.483871	0.0	2.500000	54.758065	14.0	54.430380	...	0	0	0	0	1	
15	15.0	100.0	35.317460	2.380952	0.555556	0.0	2.936508	58.809524	15.0	56.227758	...	0	0	0	0	0	
16	16.0	100.0	26.250962	1.924557	0.769823	0.0	2.232487	68.822171	16.0	59.245283	...	0	0	0	0	0	
17	17.0	100.0	28.841413	2.136401	0.164339	0.0	2.382909	66.474938	17.0	49.579832	...	0	0	0	0	0	
18	18.0	100.0	31.219110	2.471170	0.329489	0.0	3.624382	62.355848	18.0	58.436214	...	0	0	0	0	0	
19	19.0	100.0	33.652313	2.791069	0.239234	0.0	3.110048	60.207337	19.0	52.791878	...	0	0	0	0	0	

19 rows × 50 columns

In [111…
```
status_all
```

Out[111]:

	survey	survey_x	status immediate independence	status immediate unification	status is already an independent country	status is already an unified country	status maintain the status quo forever	status maintain the status quo, decide either unification or independence in the future	status maintain the status quo, move toward independence in the future	status maintain the status quo, move toward unification in the future	...	10.0	11.0
0	1.1	100.0	3.996367	1.907357	14.350590	0.272480	13.533152	29.064487	24.341508	12.534060	...	0	0
1	1.2	100.0	4.761905	1.231527	0.000000	0.000000	25.533662	34.564860	20.114943	9.195402	...	0	0
2	2.0	100.0	7.481297	2.078138	0.000000	0.000000	14.713217	32.834580	27.265170	15.627598	...	0	0
3	3.0	100.0	4.390244	1.382114	0.000000	0.000000	26.260163	36.178862	18.536585	8.780488	...	0	0
4	4.0	100.0	4.142973	2.030869	0.162470	0.081235	23.639318	36.555646	19.171405	7.473599	...	0	0
5	5.0	100.0	3.470032	2.602524	0.157729	0.000000	25.315457	34.542587	16.403785	12.223975	...	0	0
6	6.0	100.0	6.150794	3.174603	0.000000	0.000000	15.575397	31.746032	26.984127	16.369048	...	0	0
7	7.0	100.0	4.512490	2.739726	0.080580	0.000000	22.481869	36.019339	16.518936	13.456890	...	0	0
9	9.0	100.0	4.530478	3.047776	0.164745	0.000000	25.453048	32.372323	14.662273	13.509061	...	0	0
10	10.0	100.0	4.333333	1.916667	0.166667	0.000000	22.833333	35.833333	14.916667	15.083333	...	1	0
11	11.0	100.0	3.937654	1.968827	0.000000	0.000000	27.235439	33.388023	15.504512	11.730927	...	0	1
13	13.0	100.0	5.934066	0.952381	0.073260	0.000000	30.329670	28.864469	22.344322	6.593407	...	0	0
14	14.0	100.0	3.467742	0.806452	0.161290	0.000000	29.193548	31.209677	22.983871	7.258065	...	0	0
15	15.0	100.0	4.126984	0.793651	0.317460	0.000000	28.095238	30.634921	26.428571	5.158730	...	0	0
16	16.0	100.0	6.928406	0.384911	0.230947	0.000000	22.247883	28.252502	32.332564	5.003849	...	0	0
17	17.0	100.0	7.559573	1.396878	0.328677	0.000000	22.925226	29.745275	29.087921	4.519310	...	0	0
18	18.0	100.0	6.754530	1.235585	0.576606	0.000000	24.876442	28.665568	27.924217	4.695222	...	0	0
19	19.0	100.0	5.342903	0.797448	0.239234	0.000000	25.598086	30.462520	25.757576	6.060606	...	0	0

18 rows × 54 columns

Visualizations

We create a correlation matrix to visualize changes in the dependent variables over time and how they correspond with the independent variables and regression plots to show the relationship between each dependent and independent variable (without controlling for any other variables).

Correlation Matrix

We next visualize the data in a correlation matrix.

We begin by creating data frames with only the relevant data, find each variables correlation, and dropping extra variables. Specifically, we are only looking at how dependent variables correlate with independent variables, so we can drop columns with independent variables and keep them as rows only.

In [112…

```
# create correlation matrix, Tsai csr

# keep only relavant columns
df_tsai_csr_correl = tsai_csr_all[['tsai_csr not satisfied at all',
                                   'tsai_csr not very satisfied',
                                   'tsai_csr satisfied',
                                   'tsai_csr very satisfied',
                                   'Neutral %',
                                   'Positive %',
                                   'text_sentiment',
                                   'world %',
                                   'taiwan %',
                                   'sports %',
                                   'front %',
                                   'editorial %',
                                   'business %',
                                   'featured %',
                                   'article_count',
                                   'topic 0',
                                   'topic 1',
                                   'topic 2',
                                   'topic 3',
                                   'topic 4',
                                   'topic 5',
                                   'topic 6',
                                   'topic 7',
                                   'topic 8',
```

```
                                    'topic 9']].copy()

# correlation
corr_matrix1 = df_tsai_csr_correl.corr()

# drop extra info
corr1 = corr_matrix1.drop(['Neutral %', 'Positive %', 'text_sentiment', 'world %', 'taiwan %', 'sports %', 'fro
         'business %','featured %','article_count','topic 0','topic 1','topic 2','topic 3','topic 4','topic 5',
         'topic 6','topic 7','topic 8','topic 9', 'editorial %',])

corr1 = corr1.drop(['tsai_csr not satisfied at all','tsai_csr not very satisfied',
                    'tsai_csr satisfied','tsai_csr very satisfied'], axis=1)
```

In [113…

```
# create correlation matrix, csr key

# keep only relavant columns
df_csr_key_correl = csr_important_all[['most_import cross-strait relations',
                                    'Neutral %',
                                    'Positive %',
                                    'text_sentiment',
                                    'world %',
                                    'taiwan %',
                                    'sports %',
                                    'front %',
                                    'editorial %',
                                    'business %',
                                    'featured %',
                                    'article_count',
                                    'topic 0',
                                    'topic 1',
                                    'topic 2',
                                    'topic 3',
                                    'topic 4',
                                    'topic 5',
                                    'topic 6',
                                    'topic 7',
                                    'topic 8',
                                    'topic 9']].copy()

# most important
corr_matrix2 = df_csr_key_correl.corr()

# drop extra info
corr2 = corr_matrix2.drop(['Neutral %', 'Positive %', 'text_sentiment', 'world %', 'taiwan %', 'sports %', 'fro
         'business %','featured %','article_count','topic 0','topic 1','topic 2','topic 3','topic 4','topic 5',
         'topic 6','topic 7','topic 8','topic 9', 'editorial %',])

corr2 = corr2.drop(['most_import cross-strait relations'], axis=1)
```

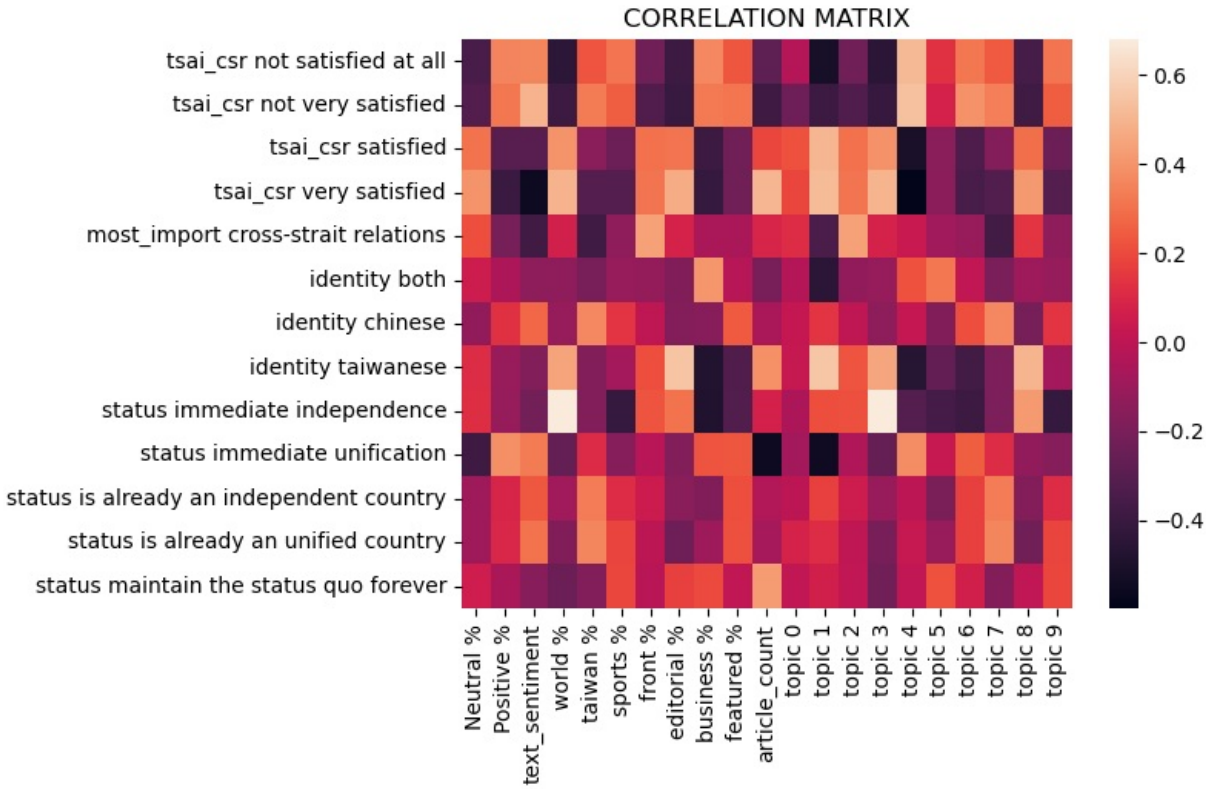
In [114…

```
# create correlation matrix, identity

# keep only relavant columns
df_identity_correl = identity_all[['identity both',
                                    'identity chinese',
                                    'identity taiwanese',
                                    'Neutral %',
                                    'Positive %',
                                    'text_sentiment',
                                    'world %',
                                    'taiwan %',
                                    'sports %',
                                    'front %',
                                    'editorial %',
                                    'business %',
                                    'featured %',
                                    'article_count',
                                    'topic 0',
                                    'topic 1',
                                    'topic 2',
                                    'topic 3',
                                    'topic 4',
                                    'topic 5',
                                    'topic 6',
                                    'topic 7',
                                    'topic 8',
                                    'topic 9']].copy()

# correlation
corr_matrix3 = df_identity_correl.corr()

# drop extra info
corr3 = corr_matrix3.drop(['Neutral %', 'Positive %', 'text_sentiment', 'world %', 'taiwan %', 'sports %', 'fro
         'business %','featured %','article_count','topic 0','topic 1','topic 2','topic 3','topic 4','topic 5',
         'topic 6','topic 7','topic 8','topic 9', 'editorial %',])

corr3 = corr3.drop(['identity both',
                    'identity chinese',
                    'identity taiwanese',], axis=1)
```

In [115…
```
# create correlation matrix, status

# keep only relavant columns
df_status_correl = status_all[['status immediate independence',
                           'status immediate unification',
                           'status is already an independent country',
                           'status is already an unified country',
                           'status maintain the status quo forever',
                           'Neutral %',
                           'Positive %',
                           'text_sentiment',
                           'world %',
                           'taiwan %',
                           'sports %',
                           'front %',
                           'editorial %',
                           'business %',
                           'featured %',
                           'article_count',
                           'topic 0',
                           'topic 1',
                           'topic 2',
                           'topic 3',
                           'topic 4',
                           'topic 5',
                           'topic 6',
                           'topic 7',
                           'topic 8',
                           'topic 9']].copy()

# correlation
corr_matrix4 = df_status_correl.corr()

# drop extra info
corr4 = corr_matrix4.drop(['Neutral %', 'Positive %', 'text_sentiment', 'world %', 'taiwan %', 'sports %', 'fro
         'business %','featured %','article_count','topic 0','topic 1','topic 2','topic 3','topic 4','topic 5',
         'topic 6','topic 7','topic 8','topic 9', 'editorial %',])

corr4 = corr4.drop(['status immediate independence',
                    'status immediate unification',
                    'status is already an independent country',
                    'status is already an unified country',
                    'status maintain the status quo forever'], axis=1)
```

The matrix output is shown below. On the left, we have each dependent variable and on the bottom each independent variable. The results are quite interesting. For example, Tsai's cross-strait relations evaluations heavily correspond with topics 1, 3, and 4. Whether cross-strait relations is the most important issue corresponds with front page *Taipei Times* stories which makes sense if salient China-related events are dictating respondent's answers to survey questions. Identifying as Taiwanese corresponds negatively with topic 4, suggesting topic 4 may relate to positive perceptions of China. Finally, variables related to unification include several article classifications and topic 7.

In [116…
```
# Matrix Visualization

# combine dataframes
corr_matrix = pd.concat([corr1, corr2, corr3, corr4])

# visualize
sns.heatmap(corr_matrix, annot=False)
plt.title("CORRELATION MATRIX")
plt.show()
```



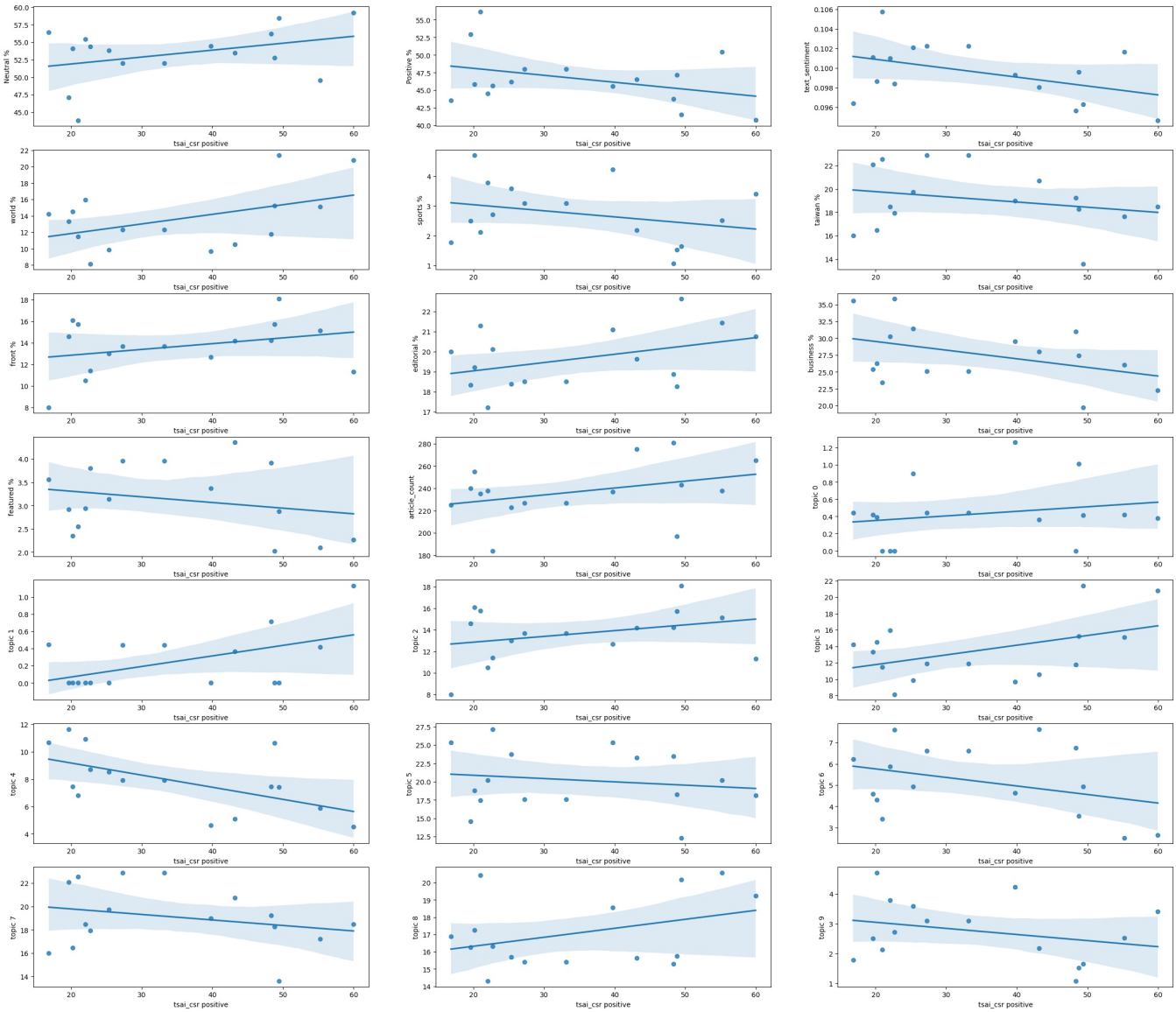
Regression Visualization

We finally analyze the aggregated data with regression charts. We create 21 plots for each dependent variable, for the 21 external variation-related independent variables.

The visualizations are shown below. While they provide interesting visualizations, they do not control for any other variables so they can only provide us with so much information.

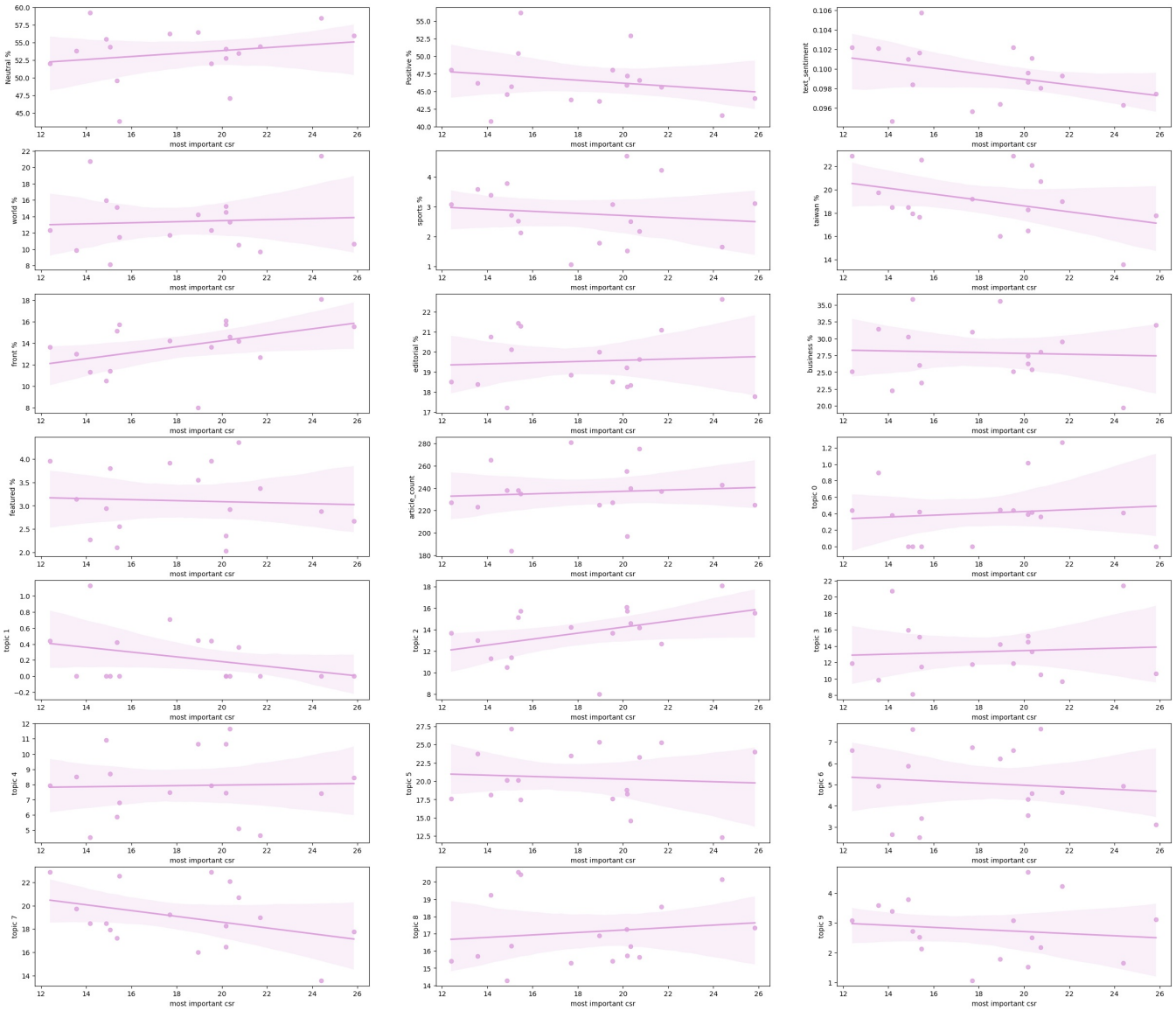
In [117…

```
# Tsi CSR

# combine positive evaluations
tsai_csr_all['tsai_csr positive'] = tsai_csr_all['tsai_csr satisfied'] + tsai_csr_all['tsai_csr very satisfied'

# plot scatterplots
fig, axes = plt.subplots(7, 3, figsize=(30, 26))

sns.regplot(ax=axes[0, 0], data=tsai_csr_all, x='tsai_csr positive', y='Neutral %')
sns.regplot(ax=axes[0, 1], data=tsai_csr_all, x='tsai_csr positive', y='Positive %')
sns.regplot(ax=axes[0, 2], data=tsai_csr_all, x='tsai_csr positive', y='text_sentiment')
sns.regplot(ax=axes[1, 0], data=tsai_csr_all, x='tsai_csr positive', y='world %')
sns.regplot(ax=axes[1, 1], data=tsai_csr_all, x='tsai_csr positive', y='sports %')
sns.regplot(ax=axes[1, 2], data=tsai_csr_all, x='tsai_csr positive', y='taiwan %')
sns.regplot(ax=axes[2, 0], data=tsai_csr_all, x='tsai_csr positive', y='front %')
sns.regplot(ax=axes[2, 1], data=tsai_csr_all, x='tsai_csr positive', y='editorial %')
sns.regplot(ax=axes[2, 2], data=tsai_csr_all, x='tsai_csr positive', y='business %')
sns.regplot(ax=axes[3, 0], data=tsai_csr_all, x='tsai_csr positive', y='featured %')
sns.regplot(ax=axes[3, 1], data=tsai_csr_all, x='tsai_csr positive', y='article_count')
sns.regplot(ax=axes[3, 2], data=tsai_csr_all, x='tsai_csr positive', y='topic 0')
sns.regplot(ax=axes[4, 0], data=tsai_csr_all, x='tsai_csr positive', y='topic 1')
sns.regplot(ax=axes[4, 1], data=tsai_csr_all, x='tsai_csr positive', y='topic 2')
sns.regplot(ax=axes[4, 2], data=tsai_csr_all, x='tsai_csr positive', y='topic 3')
sns.regplot(ax=axes[5, 0], data=tsai_csr_all, x='tsai_csr positive', y='topic 4')
sns.regplot(ax=axes[5, 1], data=tsai_csr_all, x='tsai_csr positive', y='topic 5')
sns.regplot(ax=axes[5, 2], data=tsai_csr_all, x='tsai_csr positive', y='topic 6')
sns.regplot(ax=axes[6, 0], data=tsai_csr_all, x='tsai_csr positive', y='topic 7')
sns.regplot(ax=axes[6, 1], data=tsai_csr_all, x='tsai_csr positive', y='topic 8')
sns.regplot(ax=axes[6, 2], data=tsai_csr_all, x='tsai_csr positive', y='topic 9');
```



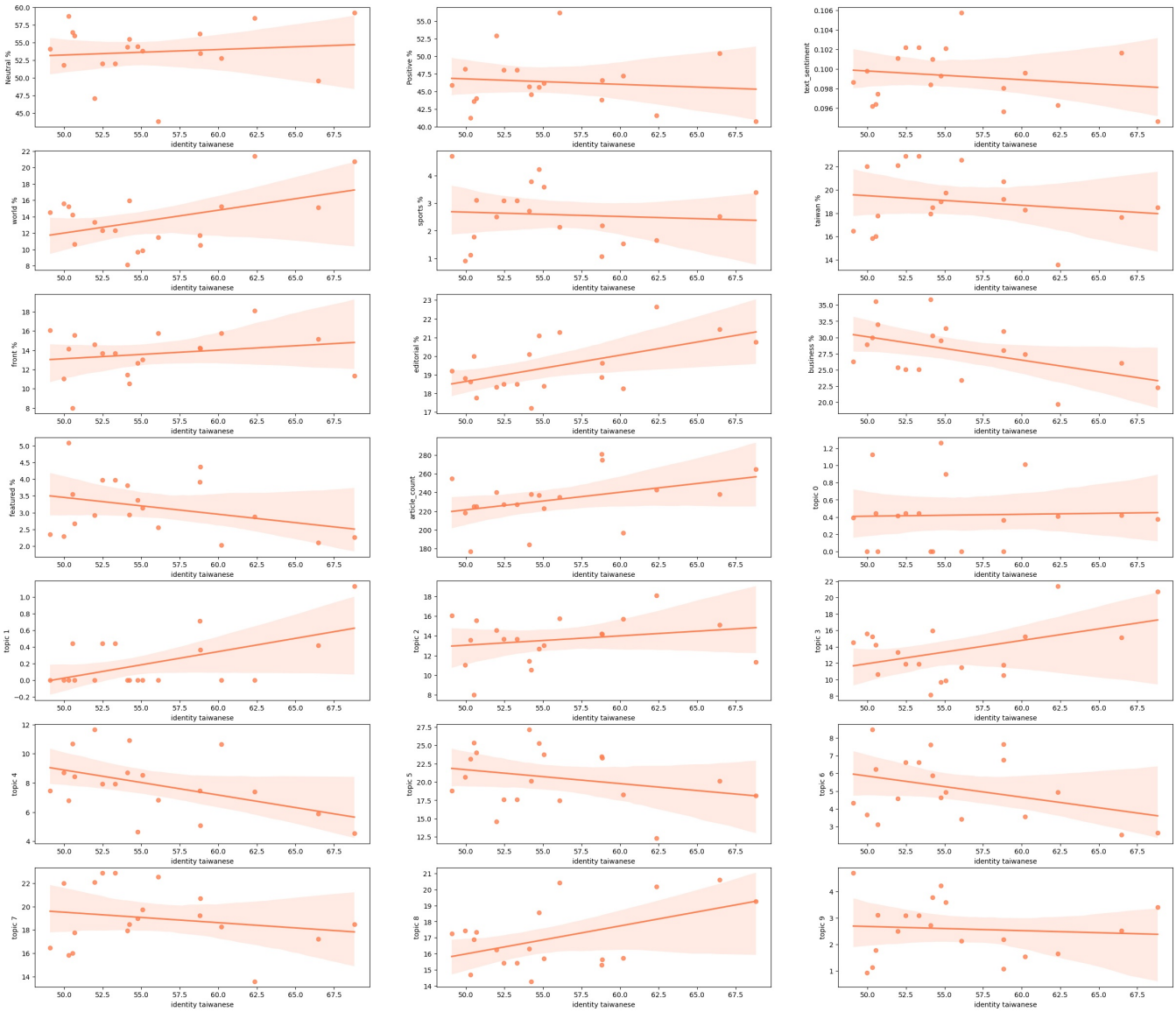
In [118…

```
# CSR most important

# combine positive evaluations
csr_important_all['most important csr'] = csr_important_all['most_import cross-strait relations']

# plot scatterplots
fig, axes = plt.subplots(7, 3, figsize=(30, 26))

sns.regplot(ax=axes[0, 0], data=csr_important_all, x='most important csr', y='Neutral %', color='plum')
sns.regplot(ax=axes[0, 1], data=csr_important_all, x='most important csr', y='Positive %', color='plum')
sns.regplot(ax=axes[0, 2], data=csr_important_all, x='most important csr', y='text_sentiment', color='plum')
sns.regplot(ax=axes[1, 0], data=csr_important_all, x='most important csr', y='world %', color='plum')
sns.regplot(ax=axes[1, 1], data=csr_important_all, x='most important csr', y='sports %', color='plum')
sns.regplot(ax=axes[1, 2], data=csr_important_all, x='most important csr', y='taiwan %', color='plum')
sns.regplot(ax=axes[2, 0], data=csr_important_all, x='most important csr', y='front %', color='plum')
sns.regplot(ax=axes[2, 1], data=csr_important_all, x='most important csr', y='editorial %', color='plum')
sns.regplot(ax=axes[2, 2], data=csr_important_all, x='most important csr', y='business %', color='plum')
sns.regplot(ax=axes[3, 0], data=csr_important_all, x='most important csr', y='featured %', color='plum')
sns.regplot(ax=axes[3, 1], data=csr_important_all, x='most important csr', y='article_count', color='plum')
sns.regplot(ax=axes[3, 2], data=csr_important_all, x='most important csr', y='topic 0', color='plum')
sns.regplot(ax=axes[4, 0], data=csr_important_all, x='most important csr', y='topic 1', color='plum')
sns.regplot(ax=axes[4, 1], data=csr_important_all, x='most important csr', y='topic 2', color='plum')
sns.regplot(ax=axes[4, 2], data=csr_important_all, x='most important csr', y='topic 3', color='plum')
sns.regplot(ax=axes[5, 0], data=csr_important_all, x='most important csr', y='topic 4', color='plum')
sns.regplot(ax=axes[5, 1], data=csr_important_all, x='most important csr', y='topic 5', color='plum')
sns.regplot(ax=axes[5, 2], data=csr_important_all, x='most important csr', y='topic 6', color='plum')
sns.regplot(ax=axes[6, 0], data=csr_important_all, x='most important csr', y='topic 7', color='plum')
sns.regplot(ax=axes[6, 1], data=csr_important_all, x='most important csr', y='topic 8', color='plum')
sns.regplot(ax=axes[6, 2], data=csr_important_all, x='most important csr', y='topic 9', color='plum');
```



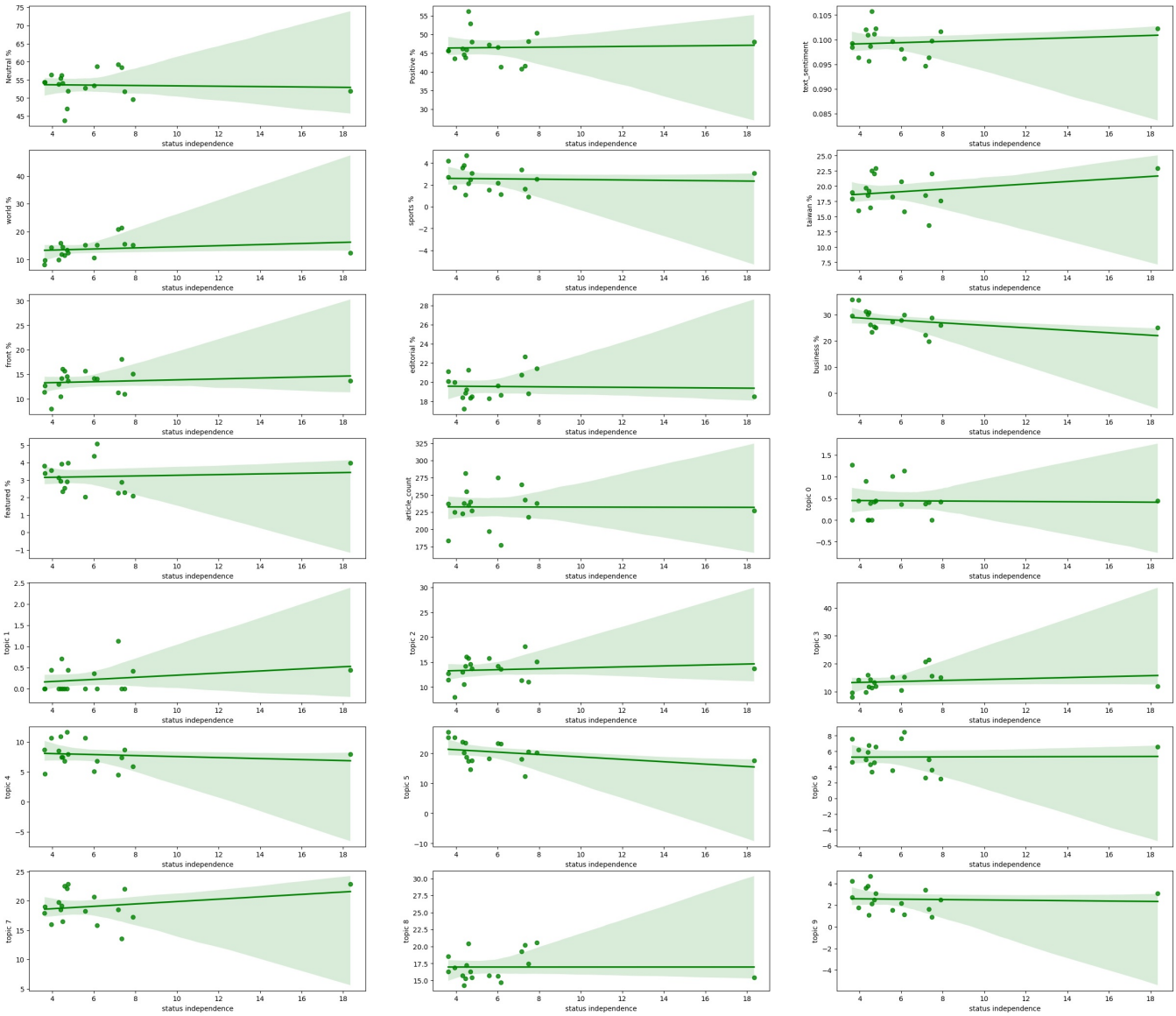
In [119…

```
# Identity

# plot scatterplots
fig, axes = plt.subplots(7, 3, figsize=(30, 26))

sns.regplot(ax=axes[0, 0], data=identity_all, x='identity taiwanese', y='Neutral %', color='coral')
sns.regplot(ax=axes[0, 1], data=identity_all, x='identity taiwanese', y='Positive %', color='coral')
sns.regplot(ax=axes[0, 2], data=identity_all, x='identity taiwanese', y='text_sentiment', color='coral')
sns.regplot(ax=axes[1, 0], data=identity_all, x='identity taiwanese', y='world %', color='coral')
sns.regplot(ax=axes[1, 1], data=identity_all, x='identity taiwanese', y='sports %', color='coral')
sns.regplot(ax=axes[1, 2], data=identity_all, x='identity taiwanese', y='taiwan %', color='coral')
sns.regplot(ax=axes[2, 0], data=identity_all, x='identity taiwanese', y='front %', color='coral')
sns.regplot(ax=axes[2, 1], data=identity_all, x='identity taiwanese', y='editorial %', color='coral')
sns.regplot(ax=axes[2, 2], data=identity_all, x='identity taiwanese', y='business %', color='coral')
sns.regplot(ax=axes[3, 0], data=identity_all, x='identity taiwanese', y='featured %', color='coral')
sns.regplot(ax=axes[3, 1], data=identity_all, x='identity taiwanese', y='article_count', color='coral')
sns.regplot(ax=axes[3, 2], data=identity_all, x='identity taiwanese', y='topic 0', color='coral')
sns.regplot(ax=axes[4, 0], data=identity_all, x='identity taiwanese', y='topic 1', color='coral')
sns.regplot(ax=axes[4, 1], data=identity_all, x='identity taiwanese', y='topic 2', color='coral')
sns.regplot(ax=axes[4, 2], data=identity_all, x='identity taiwanese', y='topic 3', color='coral')
sns.regplot(ax=axes[5, 0], data=identity_all, x='identity taiwanese', y='topic 4', color='coral')
sns.regplot(ax=axes[5, 1], data=identity_all, x='identity taiwanese', y='topic 5', color='coral')
sns.regplot(ax=axes[5, 2], data=identity_all, x='identity taiwanese', y='topic 6', color='coral')
sns.regplot(ax=axes[6, 0], data=identity_all, x='identity taiwanese', y='topic 7', color='coral')
sns.regplot(ax=axes[6, 1], data=identity_all, x='identity taiwanese', y='topic 8', color='coral')
sns.regplot(ax=axes[6, 2], data=identity_all, x='identity taiwanese', y='topic 9', color='coral');
```



In [120…

```
# Status

# combine independence evaluations
status_all['status independence'] = status_all['status immediate independence'] + status_all['status is already

# plot scatterplots
fig, axes = plt.subplots(7, 3, figsize=(30, 26))

sns.regplot(ax=axes[0, 0], data=status_all, x='status independence', y='Neutral %', color='green')
sns.regplot(ax=axes[0, 1], data=status_all, x='status independence', y='Positive %', color='green')
sns.regplot(ax=axes[0, 2], data=status_all, x='status independence', y='text_sentiment', color='green')
sns.regplot(ax=axes[1, 0], data=status_all, x='status independence', y='world %', color='green')
sns.regplot(ax=axes[1, 1], data=status_all, x='status independence', y='sports %', color='green')
sns.regplot(ax=axes[1, 2], data=status_all, x='status independence', y='taiwan %', color='green')
sns.regplot(ax=axes[2, 0], data=status_all, x='status independence', y='front %', color='green')
sns.regplot(ax=axes[2, 1], data=status_all, x='status independence', y='editorial %', color='green')
sns.regplot(ax=axes[2, 2], data=status_all, x='status independence', y='business %', color='green')
sns.regplot(ax=axes[3, 0], data=status_all, x='status independence', y='featured %', color='green')
sns.regplot(ax=axes[3, 1], data=status_all, x='status independence', y='article_count', color='green')
sns.regplot(ax=axes[3, 2], data=status_all, x='status independence', y='topic 0', color='green')
sns.regplot(ax=axes[4, 0], data=status_all, x='status independence', y='topic 1', color='green')
sns.regplot(ax=axes[4, 1], data=status_all, x='status independence', y='topic 2', color='green')
sns.regplot(ax=axes[4, 2], data=status_all, x='status independence', y='topic 3', color='green')
sns.regplot(ax=axes[5, 0], data=status_all, x='status independence', y='topic 4', color='green')
sns.regplot(ax=axes[5, 1], data=status_all, x='status independence', y='topic 5', color='green')
sns.regplot(ax=axes[5, 2], data=status_all, x='status independence', y='topic 6', color='green')
sns.regplot(ax=axes[6, 0], data=status_all, x='status independence', y='topic 7', color='green')
sns.regplot(ax=axes[6, 1], data=status_all, x='status independence', y='topic 8', color='green')
sns.regplot(ax=axes[6, 2], data=status_all, x='status independence', y='topic 9', color='green');
```



OLS Regression

Finally, we use panel data to run OLS regression. Unfortunately, if we used the aggregated data there would likely be heavily skewed results because there are only twenty surveys and over twenty independent variables, which would cause negative degrees of freedom and lots of issues with our regression results. Thus, we use panel data and attach the aggregate measures of external variation for each survey. The results are far from perfect but we are not attempting to provide a casual explanation.

We start by combining the panel data and external variation data.

In [121…
```
# combine surveys
df_panel = pd.merge(df_survey_clean, external_variation_df, left_on='survey', right_on='survey')
```

We create dummy variables again for each variable that we are using. Most notably, we create dummy variables for satisfication with Tsai's cross-strait relations policy, for cross-strait relations as the most important issue, for those who identify as Taiwanese, and those in favor of independence or believe that the status quo is independent.

We find these dummy variables primarily by creating a column full of zero's then adding a one if the neccessary condition is met.

In [122…
```
# create variables for regression

# Tsai CSR
df_panel['tsai_csr'] = df_panel['tsai_csr'].fillna('no answer')
df_panel['Tsai_CSR_pos'] = 0
df_panel.loc[df_panel.tsai_csr.str.contains('tsai_csr satisfied|tsai_csr very satisfied'),'Tsai_CSR_pos'] = 1

# Most Important
df_panel['most_import'] = df_panel['most_import'].fillna('no answer')
df_panel['CSR_first'] = [1 if x == 'most_import cross-strait relations' else 0 for x in df_panel['most_import']]
df_panel = df_panel.drop(['most_import'], axis=1)

# identity
df_panel['identity'] = df_panel['identity'].fillna('no answer')
df_panel['Identity_Taiwanese'] = 0
df_panel.loc[df_panel.identity.str.contains('identity taiwanese'),'Identity_Taiwanese'] = 1
```

```
# status
df_panel['status'] = df_panel['status'].fillna('no answer')
df_panel['Status_Independent'] = 0
df_panel.loc[df_panel.status.str.contains('status immediate independence|status is already an independent count

# gender
df_panel['gender'] = df_panel['gender'].fillna('no answer')
df_panel['Female'] = 0
df_panel.loc[df_panel.gender.str.contains('female'),'Female'] = 1

# party
df_panel['party'] = df_panel['party'].fillna('no answer')
df_panel['DPP'] = 0
df_panel.loc[df_panel.party.str.contains('dpp'),'DPP'] = 1

df_panel['KMT'] = 0
df_panel.loc[df_panel.party.str.contains('kmt'),'KMT'] = 1

# tsai_perf
df_panel['tsai_perf'] = df_panel['tsai_perf'].fillna('no answer')
df_panel['Tsai_Perf_Pos'] = 0
df_panel.loc[df_panel.tsai_perf.str.contains('satisfied|very satisfied'),'Tsai_Perf_Pos'] = 1

# past_econ
df_panel['past_econ'] = df_panel['past_econ'].fillna('no answer')
df_panel['Past_Econ_Pos'] = 0
df_panel.loc[df_panel.past_econ.str.contains('better'),'Past_Econ_Pos'] = 1

# fut_econ
df_panel['fut_econ'] = df_panel['fut_econ'].fillna('no answer')
df_panel['Fut_Econ_Pos'] = 0
df_panel.loc[df_panel.fut_econ.str.contains('better'),'Fut_Econ_Pos'] = 1

# education
df_panel['education'] = df_panel['education'].fillna('no answer')
df_panel['edu_college'] = 0
df_panel.loc[df_panel.education.str.contains('post-graduate education|university graduate|technical college gra

# location
df_panel['location'] = df_panel['location'].fillna('no answer')
df_panel['Taipei & New Taipei'] = 0
df_panel.loc[df_panel.location.str.contains('taipei city'),'Taipei'] = 1

df_panel['Tainan'] = 0
df_panel.loc[df_panel.location.str.contains('tainan city'),'Tainan'] = 1

df_panel['Taichung'] = 0
df_panel.loc[df_panel.location.str.contains('taichung city'),'Taichung'] = 1

df_panel['Keelung'] = 0
df_panel.loc[df_panel.location.str.contains('keelung city'),'Keelung'] = 1

# vote
df_panel['vote'] = df_panel['vote'].fillna('no answer')
df_panel['Vote_DPP'] = 0
df_panel.loc[df_panel.vote.str.contains('tsai ing-wen and chen chien-jen|tsai, ing-wen and ching-te lai'),'Vote
```

Model 1: Control Model

We begin by running a control model for each dependent variable. Specifically, we include the independent variables gender, party affiliation (the major parties DPP and KMT), evaluations of Tsai, perceptions of economic conditions, education, and the major cities in Taiwan including Taipei & New Taipei, Tainan, Taichung, and Keelung. We also include whether the respondent voted for the DPP (the winning party) in either of the presidential elections.

We then run each regression and find the residuals. We save both and visualize them later in the project.

In [123…
```
X = df_panel[['Female', 'DPP', 'KMT', 'Tsai_Perf_Pos', 'Past_Econ_Pos', 'Fut_Econ_Pos', 'edu_college', 'Taipei
              'Tainan', 'Taichung', 'Keelung', 'Vote_DPP']]
y = df_panel[['Tsai_CSR_pos']]
X = sm.add_constant(X)
est = sm.OLS(y, X).fit()

# residuals
influence = est.get_influence()
standardized_residuals = influence.resid_studentized_internal
```

In [124…
```
X = df_panel[['Female', 'DPP', 'KMT', 'Tsai_Perf_Pos', 'Past_Econ_Pos', 'Fut_Econ_Pos',
              'edu_college', 'Taipei & New Taipei', 'Tainan',
          'Taichung', 'Keelung', 'Vote_DPP']]
y = df_panel[['CSR_first']]
X = sm.add_constant(X)
est1 = sm.OLS(y, X).fit()

# residuals
influence1 = est1.get_influence()
```

```
standardized_residuals1 = influence1.resid_studentized_internal
```

In [125…
```
X = df_panel[['Female', 'DPP', 'KMT', 'Tsai_Perf_Pos', 'Past_Econ_Pos', 'Fut_Econ_Pos',
              'edu_college', 'Taipei & New Taipei', 'Tainan',
           'Taichung', 'Keelung', 'Vote_DPP']]
y = df_panel[['Identity_Taiwanese']]
X = sm.add_constant(X)
est2 = sm.OLS(y, X).fit()

# residuals
influence2 = est2.get_influence()
standardized_residuals2 = influence2.resid_studentized_internal
```

In [126…
```
X = df_panel[['Female', 'DPP', 'KMT', 'Tsai_Perf_Pos', 'Past_Econ_Pos', 'Fut_Econ_Pos',
              'edu_college', 'Taipei & New Taipei', 'Tainan',
           'Taichung', 'Keelung', 'Vote_DPP']]
y = df_panel[['Status_Independent']]
X = sm.add_constant(X)
est3 = sm.OLS(y, X).fit()

# residuals
influence3 = est3.get_influence()
standardized_residuals3 = influence3.resid_studentized_internal
```

The first series of regression models is visualized below. For the most part, the results make sense. The results here are mostly to control for in-survey factors so that we can see how the remaining variation is impacted by external survey factors.

To visualize the results, we use the Stargazer package which allows us to create an html rendering of our regression tables.

In [127…
```
# control variables regression results
stargazer = Stargazer([est, est1, est2, est3])

HTML(stargazer.render_html())
```

Out[127]:

	(1)	(2)	(3)	(4)
DPP	0.277***	-0.009	0.224***	0.048***
	(0.006)	(0.006)	(0.008)	(0.004)
Female	0.044***	0.008	0.030***	-0.004
	(0.005)	(0.005)	(0.006)	(0.003)
Fut_Econ_Pos	0.153***	0.001	0.088***	0.024***
	(0.008)	(0.008)	(0.010)	(0.005)
KMT	-0.146***	0.136***	-0.180***	-0.025***
	(0.006)	(0.006)	(0.008)	(0.004)
Keelung	-0.007	0.005	-0.023	0.016
	(0.021)	(0.020)	(0.026)	(0.013)
Past_Econ_Pos	0.184***	0.005	0.099***	0.026***
	(0.009)	(0.009)	(0.011)	(0.006)
Taichung	-0.001	0.004	0.030***	0.005
	(0.008)	(0.008)	(0.009)	(0.005)
Tainan	0.023***	-0.009	0.070***	0.009*
	(0.009)	(0.008)	(0.011)	(0.005)
Taipei & New Taipei	0.000***	-0.000	-0.000**	0.000
	(0.000)	(0.000)	(0.000)	(0.000)
Tsai_Perf_Pos	0.168***	0.041***	-0.052***	0.014***
	(0.006)	(0.006)	(0.008)	(0.004)
Vote_DPP	0.175***	-0.046***	0.205***	0.026***
	(0.006)	(0.006)	(0.007)	(0.003)
const	0.021***	0.112***	0.473***	0.030***
	(0.007)	(0.007)	(0.009)	(0.004)
edu_college	-0.044***	-0.009*	-0.036***	-0.009***
	(0.005)	(0.005)	(0.006)	(0.003)
Observations	24,189	24,189	24,189	24,189
R^2	0.322	0.036	0.194	0.031
Adjusted R^2	0.322	0.036	0.194	0.031
Residual Std. Error	0.371 (df=24177)	0.357 (df=24177)	0.448 (df=24177)	0.223 (df=24177)
F Statistic	1045.294*** (df=11; 24177)	82.921*** (df=11; 24177)	529.212*** (df=11; 24177)	71.230*** (df=11; 24177)
Note:				*p<0.1; **p<0.05; ***p<0.01

Below we create a data frame keeping the residuals for each dependent variable.

In [128…

```
# residuals df
df_residuals = pd.DataFrame(standardized_residuals, columns = ['Residuals'])
df_residuals['Dependent Variable'] = 'Tsai_CSR'

df_residuals1 = pd.DataFrame(standardized_residuals1, columns = ['Residuals'])
df_residuals1['Dependent Variable'] = 'CSR first'

df_residuals2 = pd.DataFrame(standardized_residuals2, columns = ['Residuals'])
df_residuals2['Dependent Variable'] = 'Identity Taiwanese'

df_residuals3 = pd.DataFrame(standardized_residuals3, columns = ['Residuals'])
df_residuals3['Dependent Variable'] = 'Status Independence'

# combine dataframes
df_residuals_all = pd.concat([df_residuals, df_residuals1, df_residuals2, df_residuals3])

df_residuals_all.head()
```

Out[128]:

	Residuals	Dependent Variable
0	-0.118603	Tsai_CSR
1	-0.865019	Tsai_CSR
2	-0.573192	Tsai_CSR
3	0.000529	Tsai_CSR
4	-0.392615	Tsai_CSR

The residuals are visualized in the strip plot below. The chart shows how the distribution of the residuals for each dependent variable. It is evidence that there is a lot of variation not accounted for by the regressions above, particularly in the first and third dependent variable.

In [129…

```
sns.set(rc={'figure.figsize':(10,8)})
ax = sns.stripplot(y="Residuals", x="Dependent Variable", data=df_residuals_all)
ax.set_title("Model 1 - Residual Distribution: per Dependent Variable");
```

Model 2: Including External-Variation Measures

In the next series of models, we include measures of external variation. Specifically, we include measures of average sentiment per survey, article count per survey, article classification count per survey, and article topic type per survey. Once again, we keep the model residuals and results to show later.

Note, we included world and topic 0 as baselines for the article classification and article topic dummy variables.

In [130…

```
X = df_panel[['Female', 'DPP', 'KMT', 'Tsai_Perf_Pos', 'Past_Econ_Pos', 'Fut_Econ_Pos',
              'edu_college', 'Taipei & New Taipei', 'Tainan',
           'Taichung', 'Keelung', 'Vote_DPP', 'text_sentiment', 'taiwan %', 'sports %',
              'front %', 'editorial %', 'business %', 'featured %', 'article_count', 'topic 1', 'topic 2', 'top
              'topic 4', 'topic 5', 'topic 6', 'topic 7', 'topic 8', 'topic 9']]
y = df_panel[['Tsai_CSR_pos']]
X = sm.add_constant(X)
est4 = sm.OLS(y, X).fit()

# residuals
influence = est4.get_influence()
standardized_residuals = influence.resid_studentized_internal
```

In [131…

```
X = df_panel[['Female', 'DPP', 'KMT', 'Tsai_Perf_Pos', 'Past_Econ_Pos', 'Fut_Econ_Pos',
```

```
            'edu_college', 'Taipei & New Taipei', 'Tainan',
          'Taichung', 'Keelung', 'Vote_DPP', 'text_sentiment', 'taiwan %', 'sports %',
            'front %', 'editorial %', 'business %', 'featured %', 'article_count', 'topic 1', 'topic 2', 'top
            'topic 4', 'topic 5', 'topic 6', 'topic 7', 'topic 8', 'topic 9']]
y = df_panel[['CSR_first']]
X = sm.add_constant(X)
est5 = sm.OLS(y, X).fit()

# residuals
influence1 = est5.get_influence()
standardized_residuals1 = influence1.resid_studentized_internal
```

In [132…

```
X = df_panel[['Female', 'DPP', 'KMT', 'Tsai_Perf_Pos', 'Past_Econ_Pos', 'Fut_Econ_Pos',
            'edu_college', 'Taipei & New Taipei', 'Tainan',
          'Taichung', 'Keelung', 'Vote_DPP', 'text_sentiment', 'taiwan %', 'sports %',
            'front %', 'editorial %', 'business %', 'featured %', 'article_count', 'topic 1', 'topic 2', 'top
            'topic 4', 'topic 5', 'topic 6', 'topic 7', 'topic 8', 'topic 9']]
y = df_panel[['Identity_Taiwanese']]
X = sm.add_constant(X)
est6 = sm.OLS(y, X).fit()

# residuals
influence2 = est6.get_influence()
standardized_residuals2 = influence2.resid_studentized_internal
```

In [133…

```
X = df_panel[['Female', 'DPP', 'KMT', 'Tsai_Perf_Pos', 'Past_Econ_Pos', 'Fut_Econ_Pos',
            'edu_college', 'Taipei & New Taipei', 'Tainan',
          'Taichung', 'Keelung', 'Vote_DPP', 'text_sentiment', 'taiwan %', 'sports %',
            'front %', 'editorial %', 'business %', 'featured %', 'article_count', 'topic 1', 'topic 2', 'top
            'topic 4', 'topic 5', 'topic 6', 'topic 7', 'topic 8', 'topic 9']]
y = df_panel[['Status_Independent']]
X = sm.add_constant(X)
est7 = sm.OLS(y, X).fit()

# residuals
influence3 = est7.get_influence()
standardized_residuals3 = influence3.resid_studentized_internal
```

The external variation model series are shown below. The adjusted R-squared is higher for each model, suggesting the additional independent variables help explain more variation in each dependent variable.

In the first regression, in the evaluations of Tsai's cross-strait relations policy, each measurement of external variation is significant. Article count is marginally impactful and negative, suggesting more articles correspond with less positive evaluations of Tsai's cross-strait relations policy. This could mean more articles are indicative of more information indicting her policy or more articles written about China. Similarly, front page is negative, one again suggesting that more information on the front page suggests that something impactful to Taiwanese that pushes them to view Tsai's China policy negatively. The text sentiment variable is significant and high-magnitude which means more positive articles correspond with most positive evaluations of Tsai, which makes sense given the direction of the other variables. Each topic is significant and topic eight is negative with the highest magnitude. Topic eight is referencing microchips which is unsuprisingly negative because more concerns over China encroaching in Taiwan's microchip industry likely leads to fear that Tsai's China policy is insufficient.

The second regression, whether cross-strait relations is the most important issue, has fewer significant variables. Articles related to business, Taiwan, and on the front page are significant. Though Taiwan and the front page are negative, perhaps because other salient issues detract from classifying China policy as the most important issue. For example, if a front page article is written about a likely recession, respondents put economic concerns as the most important issue. Most of the topics are negative or insignificant, but topic two is significant with a high magnitude coefficient. Topic two appears related to the economic consequences of Covid which could push respondents to prioritize Chinese-Taiwanese policy when Covid was primarily a concern in China.

In the third regression, identification as Taiwanese, several of the article classifications are insignificant or negative. Interestingly, the Taiwanese section of the *Taipei Times* negatively corresponds with identification as Taiwanese and it could mean a lot of these articles are discussing topics that related to identifying as Chinese or both. Text sentiment is positive and high-magnitude, likely suggesting that more positive articles about Taiwan push Taiwanese towards identifying as Taiwanese. Every topic is negative, except topic two which is insignificant and topic one which relates to the Hong Kong protests. Unsuprisingly, the Hong Kong protests led more Taiwanese to identify with their state rather than China.

In the finally regression, desire for independence, almost every external variation variable is negative. Most likely, significant events related to China, such as positive economic and technological development, lead respondents to desire more integration with China. For example, when former Taiwanese President Ma ying-jeou incentivized more Chinese tourism to Taiwan in the early 2000s, more Taiwanese felt like the economic beneficts meant further integration was useful.

Of course, these results are far from casual. Our models are likely suffering from ommited variable biase and other latent issues that mean we must be cautious in our interpretation. However, the results are interesting and make sense given the theory behind our results.

In [134…

```
stargazer = Stargazer([est4, est5, est6, est7])

HTML(stargazer.render_html())
```

Out[134]:

	(1)	(2)	(3)	(4)
DPP	0.234***	0.005	0.173***	0.039***
	(0.006)	(0.006)	(0.008)	(0.004)
Female	0.006	-0.002	0.013**	-0.002
	(0.005)	(0.005)	(0.006)	(0.003)
Fut_Econ_Pos	0.122***	-0.012	0.069***	0.022***
	(0.008)	(0.008)	(0.010)	(0.005)
KMT	-0.148***	0.144***	-0.221***	-0.034***
	(0.006)	(0.006)	(0.008)	(0.004)
Keelung	-0.013	0.005	-0.043*	0.015
	(0.020)	(0.020)	(0.025)	(0.013)
Past_Econ_Pos	0.161***	-0.008	0.081***	0.027***
	(0.009)	(0.009)	(0.011)	(0.006)
Taichung	-0.008	0.003	0.010	0.002
	(0.007)	(0.007)	(0.009)	(0.005)
Tainan	0.014	-0.011	0.050***	0.007
	(0.008)	(0.008)	(0.010)	(0.005)
Taipei & New Taipei	0.000***	0.000***	0.000***	-0.000
	(0.000)	(0.000)	(0.000)	(0.000)
Tsai_Perf_Pos	0.097***	0.023***	-0.067***	0.018***
	(0.007)	(0.007)	(0.009)	(0.005)
Vote_DPP	0.194***	-0.061***	0.228***	0.029***
	(0.006)	(0.006)	(0.007)	(0.004)
article_count	-0.001***	0.000*	-0.000**	-0.001***
	(0.000)	(0.000)	(0.000)	(0.000)
business %	0.068***	0.111***	0.181***	-0.005
	(0.013)	(0.013)	(0.016)	(0.008)
const	-16.161***	1.717	-0.815	6.615***
	(2.357)	(2.360)	(2.933)	(1.484)
editorial %	0.380***	0.043*	0.145***	-0.075***
	(0.025)	(0.025)	(0.031)	(0.016)
edu_college	-0.012**	0.004	-0.022***	-0.012***
	(0.005)	(0.005)	(0.006)	(0.003)
featured %	0.242***	0.179***	0.049	-0.053***
	(0.029)	(0.029)	(0.037)	(0.019)
front %	-0.242***	-0.645***	-0.204***	-0.125***
	(0.051)	(0.051)	(0.064)	(0.032)
sports %	0.070***	-0.015	-0.026*	-0.034***
	(0.012)	(0.012)	(0.015)	(0.008)
taiwan %	0.204***	-0.177***	-0.252***	0.039
	(0.047)	(0.047)	(0.058)	(0.030)
text_sentiment	18.164***	12.937***	25.658***	-0.658

	(2.833)	(2.836)	(3.525)	(1.784)
topic 1	0.171***	-0.034*	-0.103***	-0.030**
	(0.019)	(0.019)	(0.024)	(0.012)
topic 2	0.406***	0.629***	0.193***	0.062**
	(0.049)	(0.049)	(0.061)	(0.031)
topic 3	0.160***	-0.023	0.013	-0.059***
	(0.023)	(0.023)	(0.029)	(0.014)
topic 4	0.066***	-0.125***	-0.197***	-0.066***
	(0.020)	(0.020)	(0.025)	(0.013)
topic 5	0.077***	-0.136***	-0.185***	-0.060***
	(0.020)	(0.020)	(0.025)	(0.013)
topic 6	-0.155***	-0.134***	-0.127***	0.007
	(0.010)	(0.010)	(0.013)	(0.006)
topic 7	-0.071	0.132***	0.246***	-0.098***
	(0.051)	(0.051)	(0.064)	(0.032)
topic 8	-0.238***	-0.087***	-0.170***	0.004
	(0.009)	(0.009)	(0.011)	(0.006)
topic 9	0.070***	-0.015	-0.026*	-0.034***
	(0.012)	(0.012)	(0.015)	(0.008)
Observations	24,189	24,189	24,189	24,189
R^2	0.391	0.064	0.230	0.043
Adjusted R^2	0.391	0.063	0.229	0.042
Residual Std. Error	0.352 (df=24161)	0.352 (df=24161)	0.438 (df=24161)	0.222 (df=24161)
F Statistic	575.309*** (df=27; 24161)	60.741*** (df=27; 24161)	267.498*** (df=27; 24161)	40.541*** (df=27; 24161)
Note:				*p<0.1; **p<0.05; ***p<0.01

We put all of the residuals in a data frame so that we can analyze the results per dependent variable.

In [135…
```
# residuals df
df_residuals = pd.DataFrame(standardized_residuals, columns = ['Residuals'])
df_residuals['Dependent Variable'] = 'Tsai_CSR'

df_residuals1 = pd.DataFrame(standardized_residuals1, columns = ['Residuals'])
df_residuals1['Dependent Variable'] = 'CSR first'

df_residuals2 = pd.DataFrame(standardized_residuals2, columns = ['Residuals'])
df_residuals2['Dependent Variable'] = 'Identity Taiwanese'

df_residuals3 = pd.DataFrame(standardized_residuals3, columns = ['Residuals'])
df_residuals3['Dependent Variable'] = 'Status Independence'

# combine dataframes
df_residuals_all = pd.concat([df_residuals, df_residuals1, df_residuals2, df_residuals3])

df_residuals_all.head()
```

Out[135]:

	Residuals	Dependent Variable
0	-0.087162	Tsai_CSR
1	-1.043684	Tsai_CSR
2	-0.546313	Tsai_CSR
3	-0.071403	Tsai_CSR
4	-0.491690	Tsai_CSR

The final residual results are available in the figure below. The residuals follow the same pattern as the chart above, yet there is less

variation unaccounted for in each model.

In [136…

```
sns.set(rc={'figure.figsize':(10,8)})
ax = sns.stripplot(y="Residuals", x="Dependent Variable", data=df_residuals_all)
ax.set_title("Model 2 - Residual Distribution: per Dependent Variable");
```

Conclusion

Sources

- https://international.thenewslens.com/article/129804
- https://www.taiwannews.com.tw/en/news/3812154
- https://www.theguardian.com/world/2022/mar/10/hong-kong-protests-documentary-breaks-taiwan-box-office-record-in-opening-weeks
- https://www.reuters.com/world/china/taiwan-calls-chinas-covid-lockdowns-cruel-says-wont-follow-its-steps-2022-05-01/
- https://asia.nikkei.com/Business/Tech/Semiconductors/CHIPS-Act-leaves-chipmakers-facing-choice-between-U.S.-and-China
- https://www.bbc.co.uk/news/world-asia-62398029
- http://teds.nccu.edu.tw/intro/super_pages.php?ID=intro1
- https://geodata.lib.utexas.edu/catalog/stanford-fn648mm8787
- https://geopandas.org/en/stable/
- https://github.com/yohman/workshop-python-spatial-stats/blob/main/Spatial%20Autocorrelation.ipynb
- https://db.cec.gov.tw/ElecTable/Election/ElecTickets?dataType=tickets&typeId=ELC&subjectId=P0&legisId=00&themeId=61b4dda0ebac3332203ef3729a9a0ada&dataLevel=C&prvCode=00
- https://www.bbc.co.uk/news/world-asia-34729538
- https://countwordsfree.com/stopwords
- https://sigdelta.com/blog/text-analysis-in-pandas/#:~:text=Pandas%20is%20a%20great%20tool%20for%20the%20analysis,post%20I%E2%80%99ll%20present%20them%20on

In [ ]: