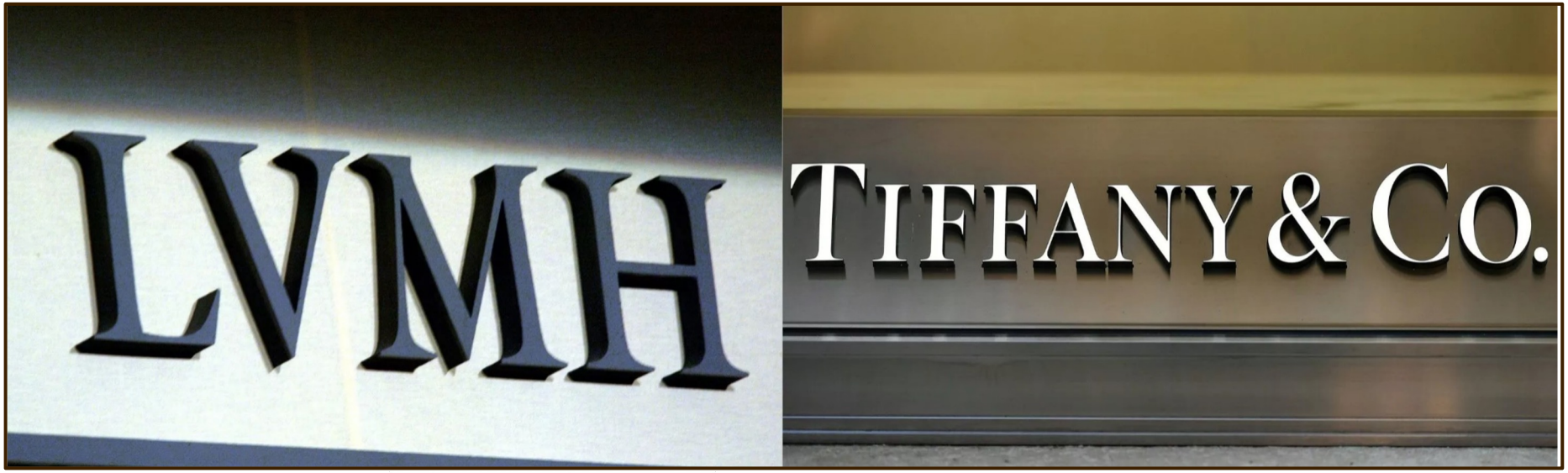
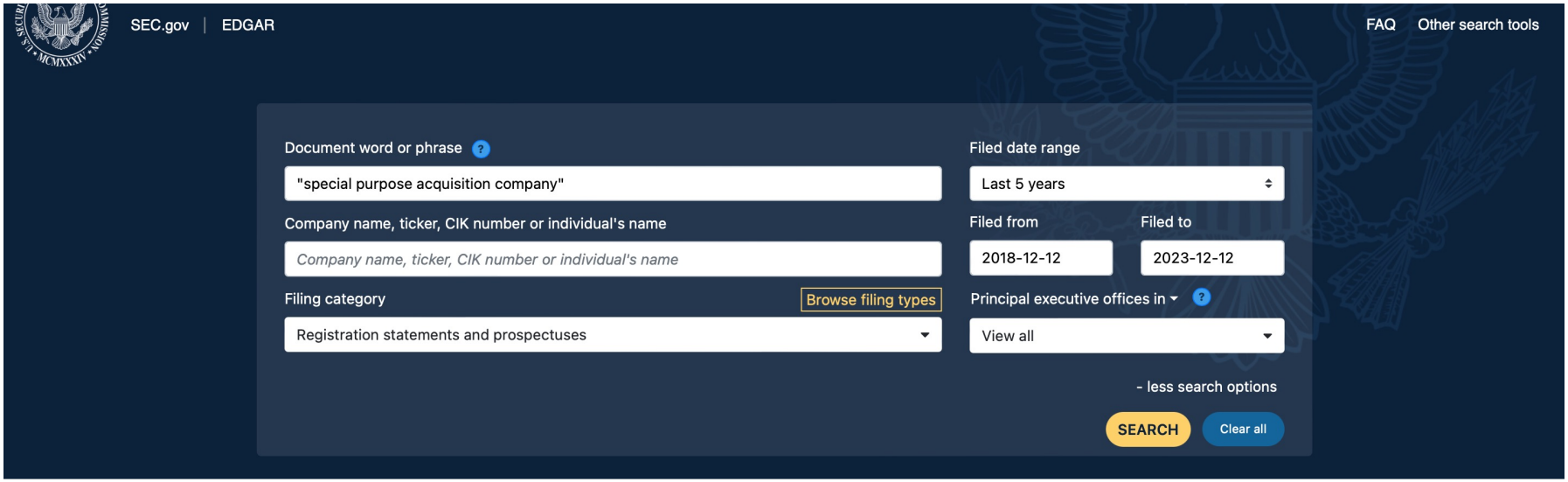


SciencesPo

SPAC size and dilution evolution

December, 12th 2023

1. Scraping EDGAR



Refine search results by:

- Entity
- Form
- Principal executive offices located in

Click headings to show top filters.
Document counts shown in #

4,270 search results

Show Columns

Filed | Reporting for | CIK | Located | Incorporated | File number | Film number

Form & File	Filed	Reporting for	Filing entity/person
S-1 (Registration statement) EX-99.4	2021-03-05		Callodine Acquisition Corp
S-1 (Registration statement)	2023-12-06		FG Merger III Corp.
S-1/A (Registration statement)	2022-01-05		Apollo Strategic Growth Capital III (APGC)
S-1/A (Registration statement)	2022-01-12		Apollo Strategic Growth Capital III (APGC)

```python
def click_s1_links(driver):
    # Create an empty list to store the results
    results = []
    num_pages = 2  # Set the number of pages to scrape

    for page in range(num_pages):
        s1_links = driver.find_elements(By.XPATH, "//a[contains(@href, 's1.htm')]")

        # Iterate through the found links and click on each of them
        for link in s1_links:
            # Extract the href attribute from the link
            href = link.get_attribute("href")
            print(f"Opening link: {href}")

            # Click on the link using JavaScript
            driver.execute_script("arguments[0].click();", link)

            # Wait for the modal to appear and handle it
            try:
                WebDriverWait(driver, 7).until(EC.visibility_of_element_located((By.ID, "previewer")))

                # Handle the modal by sending ESC key, assuming it's a dismissible overlay
                driver.find_element(By.TAG_NAME, 'body').send_keys(Keys.ESCAPE)
            except Exception as e:
                print(f"Error handling modal: {e}")

            # Add a delay to give time for the page to load or handle any pop-ups
            time.sleep(6)

            # Switch to the iframe
            try:
                iframe = driver.find_element(By.ID, "ipreviewer")  # Replace "ipreviewer" with the actual iframe ID
                driver.switch_to.frame(iframe)
                print("Switched to iframe")
```

```
        # Switch back to the default content
        driver.switch_to.default_content()
        print("Switched back to default content")
    except StaleElementReferenceException:
        print("StaleElementReferenceException: Trying to find iframe element again.")
        continue
    except Exception as e:
        print(f"Error switching to iframe: {e}")

if page < num_pages - 1:
    try:
        next_page_link = driver.find_element(By.XPATH, "//a[@data-value='nextPage' and text()='Next page']")
        driver.execute_script("arguments[0].click();", next_page_link)
        print(f"Switching to Page {page + 2}")
        time.sleep(6)
    except Exception as e:
        print(f"Error clicking Next page link: {e}")
        break
```

1. Scraping EDGAR

```
# Switch to the iframe
try:
    iframe = driver.find_element(By.ID, "ipreviewer")  # Replace "ipreviewer" with the actual iframe ID
    driver.switch_to.frame(iframe)
    print("Switched to iframe")

    # Extract the HTML content of the iframe
    iframe_html = driver.page_source

    # Use BeautifulSoup to parse the HTML content with case-insensitive search
    iframe_soup = BeautifulSoup(iframe_html, 'html.parser')

    # Define the regex pattern to find percentages based on specified phrases
    regex_pattern = re.compile(r'\b((initial\s+(?:shareholder[s]*|stockholder[s]*|Sponsors)\s*(?:\S\s*){0,70}(?:will\s+(?:collectively\s+)?(?:beneficially\s+)?own|will'

    # Use regex to find percentages in the iframe_text
    matches = regex_pattern.findall(str(iframe_soup))

    if matches:
        # Extract the percentage from the match
        percentage = matches[0][2]
        print(f"Found match: {percentage}")

        # Locate the phrase with the units and price per unit information
        text = iframe_soup.get_text()
        units_pattern = re.compile(r'Securities\s+offered[^$]*?([\d,]+)\s*units[^$]*\$\s*([\d,.]+)\s*per\s*unit')
        units_match = units_pattern.search(text)

        if units_match:
            units, price_per_unit = units_match.groups()
            print(f"Units: {units}, Price per unit: {price_per_unit}")


            #html_content = '<span class="modal-file-name">S-1 (Registration statement) of filed (2021-02-26)</span>'
            text = iframe_soup.get_text()
            date_pattern = re.compile(r'on (\w+\s+\d{1,2},\s+\d{4})')
            date_match = date_pattern.search(text)

            if date_match:
                extracted_date = date_match.group(0)
                print(f"Extracted Date: {extracted_date}")
```

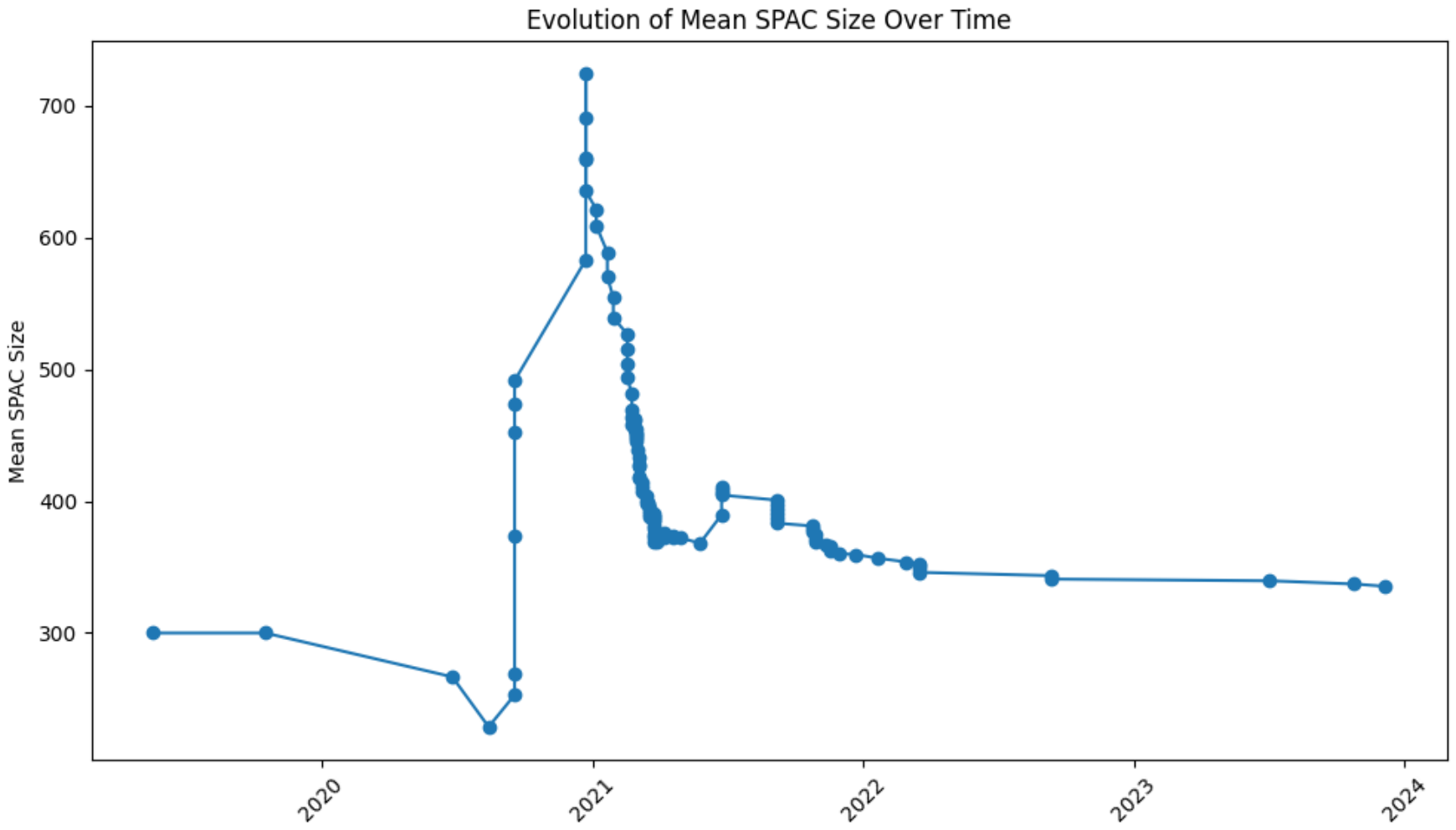
3. Dataframe Construction

```
    df = pd.DataFrame(results)

    if not df.empty:
        df['Date'] = pd.to_datetime(df['Date'])
        df = df.sort_values(by='Date')

# Reordering columns and creating 'Date SPAC' column
        df = df[['Date', 'Percentage', 'Units', 'Price Per Unit']]
        df['Date SPAC'] = df['Date'].dt.strftime('%d/%m/%Y')

# Converting 'Units' and 'Price Per Unit' to numeric
        df['Units'] = pd.to_numeric(df['Units'].str.replace(',', ''), errors='coerce')
        df['Price Per Unit'] = pd.to_numeric(df['Price Per Unit'].str.replace(',', ''), errors='coerce')

# Calculating 'SPAC size' and cumulative 'Total SPAC amount' in millions
        df['SPAC size'] = (df['Units'] * df['Price Per Unit']) / 1000000
        df['Total SPAC amount'] = df['SPAC size'].cumsum()

# Adding an 'ID' column and calculating 'Nombre de SPAC'
        df['ID'] = 1

        for test_nbr, row in df.iterrows():
          SPAC = row["SPAC size"]

          if SPAC > 0:
              df.at[test_nbr, 'ID'] = 1

          else:
              df.at[test_nbr, 'ID'] = 0

        df['Nombre de SPAC'] = df['ID'].cumsum()

        df['Mean SPAC size'] = df['Total SPAC amount'] / df['Nombre de SPAC']
        df['Mean SPAC size'] = df['Mean SPAC size'].round(1)

# Dropping the original 'Date' column
        df.drop(columns=['Date'], inplace=True)

# Reordering columns for the final output
        df = df[['Date SPAC', 'Percentage', 'SPAC size', 'Mean SPAC size']]
```

4. Data results

	Date	SPAC Percentage	SPAC size	Mean SPAC size
97	20/05/2019	20	300.0	300.0
102	18/10/2019	20	300.0	300.0
103	26/06/2020	20	200.0	266.7
104	14/08/2020	20	115.0	228.8
82	18/09/2020	20	350.0	253.0
..	...	...	...	...
63	12/09/2022	20	75.0	343.5
85	12/09/2022	20	75.0	340.9
100	03/07/2023	20	200.0	339.6
4	24/10/2023	20	75.0	337.2
0	06/12/2023	20	150.0	335.4

[109 rows x 4 columns]
Mean SPAC size: 419.10458715596326
Median SPAC size: 393.6
Standard Deviation of SPAC size: 88.99725483211297


Evolution of Mean SPAC Size Over Time
Mean SPAC Size
700
600
500
400
300
2020
2021
2022
2023
2024