A Definitive Guide to Sourcing and Visualizing Century-Long S&P 500 and Gold Data in Python

Part I: Acquiring and Preparing Long-Term S&P 500 Data

The foundation of any long-term financial analysis is the quality and temporal depth of its underlying data. For a study spanning over a century, standard financial data APIs, while convenient for recent information, often prove inadequate. They may lack the necessary historical scope or suffer from inconsistencies in how older data was compiled. Therefore, the first step in this analysis is to secure a dataset for the S&P 500 that is not only extensive but also academically vetted and widely accepted as a benchmark for historical research.

1.1 The Shiller Dataset: The Gold Standard for Historical Market Analysis

For analyzing U.S. equity market performance since the late 19th century, the premier resource is the "U.S. Stock Markets 1871-Present and CAPE Ratio" dataset, meticulously compiled and maintained by Nobel laureate Robert J. Shiller of Yale University.[1] This dataset is the bedrock of much of the long-term financial research conducted today, including its use in Shiller's seminal work, *Irrational Exuberance*.[1]

Its superiority for this specific task over other common sources is clear. For instance, the widely-used Federal Reserve Economic Data (FRED) platform provides a daily series for the S&P 500 under the ticker SP500, but its history only extends back to 2015, making it unsuitable for a century-long analysis.[3] While other APIs like yfinance can provide longer histories, they often rely on adjusted price data that can be complex to align with historical dividend and earnings information, and their data provenance can be less transparent than an academic source.

The Shiller dataset circumvents these issues by providing a single, consistent, and comprehensive file containing several key monthly series dating back to January 1871.[1] For the purpose of this report, the critical components are:

- **S&P Composite Price (P):** A monthly average of daily closing prices for the S&P Composite index (the precursor to the S&P 500).
- **Consumer Price Index (CPI):** A measure of inflation, which is essential for any real-terms analysis and is used to construct the Cyclically Adjusted Price-to-Earnings (CAPE) ratio for which the dataset is famous.[1]
- **Dividends and Earnings:** While not directly used for calculating the S&P 500 priced in gold, their inclusion makes the dataset a one-stop-shop for deep historical market analysis.[1]

The academic credibility, transparency of methodology, and direct accessibility of this dataset from the Yale University website make it the unambiguous starting point for building a robust and defensible long-term analysis.[1]

1.2 Python Implementation: Accessing and Processing Shiller's Data

A modern data analysis workflow should be reproducible and minimize manual steps. Fortunately, the Shiller dataset, provided as an Excel file, can be ingested and processed directly into a pandas DataFrame without needing to be downloaded manually. The following Python code demonstrates this process.

The script performs several key operations:

1. It specifies the direct URL to the Excel file on Professor Shiller's website.
2. It uses pandas.read_excel to load the data. Crucially, it specifies the sheet name ('Data') and the number of header rows to skip (skiprows=7) to bypass the introductory text in the spreadsheet.
3. It selects and renames the essential columns for clarity: the date column (unnamed in the original file), the S&P Composite price (P), and the Consumer Price Index (CPI).
4. It performs data cleaning by removing any rows that do not contain valid data, which often appear at the end of the spreadsheet as footnotes or summary statistics.
5. It converts the data types to their appropriate formats. The date column is processed to handle the specific YYYY.MM format used in the source file, and the price and CPI columns are converted to numeric types.
6. Finally, it sets the Date column as the DataFrame index, a best practice for time series

analysis in pandas.

Python

```python
import pandas as pd
import requests
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt
import numpy as np

# --- Part I: Acquire and Prepare S&P 500 Data ---

def get_shiller_sp500_data():
    """
    Downloads and processes Robert Shiller's long-term S&P 500 data.

    Returns:
        pandas.DataFrame: A cleaned DataFrame with monthly S&P 500 price and CPI
                          data from 1871 to the present.
    """
    # URL for Shiller's data (hosted on his Yale University page)
    url = "http://www.econ.yale.edu/~shiller/data/ie_data.xls"

    # Read the Excel file, skipping the initial header rows
    # The actual data starts on row 8 (index 7)
    try:
        df = pd.read_excel(url, sheet_name='Data', skiprows=7)
    except Exception as e:
        print(f"Failed to download or read the Excel file: {e}")
        return None

    # Select and rename the relevant columns
    # Column names are based on the structure of Shiller's spreadsheet
    df = df[['Unnamed: 0', 'P', 'CPI']].copy()
    df.rename(columns={'Unnamed: 0': 'Date', 'P': 'S&P500_Price'}, inplace=True)

    # Remove footer rows which may be read as NaN
    df.dropna(inplace=True)

    # Convert the 'Date' column to datetime objects
    # Shiller's date format is YYYY.MM (e.g., 1871.01)
    def format_date(date_val):
        if isinstance(date_val, float):
            year = int(date_val)
            # The month is represented by the fractional part
            month = int(round((date_val - year) * 100))
            # Handle cases like 1871.1 which should be October (10)
            if month == 1:
                month = 10
            return pd.to_datetime(f'{year}-{month:02d}-01')
        return pd.NaT

    df = df.apply(format_date)

    # Convert price and CPI columns to numeric types
    df = pd.to_numeric(df, errors='coerce')
    df['CPI'] = pd.to_numeric(df['CPI'], errors='coerce')

    # Set the date as the index
    df.set_index('Date', inplace=True)

    # Filter for the desired time period (e.g., from 1900 onwards)
    df_sp500 = df.loc['1900-01-01':].copy()

    print("S&P 500 data successfully processed:")
    print(df_sp500.head())
    print(df_sp500.tail())

    return df_sp500

# Execute the function
df_sp500 = get_shiller_sp500_data()
```

The output of this process is a clean, monthly pandas DataFrame containing the S&P 500 price, perfectly prepared for the subsequent steps of the analysis. This methodical approach ensures that the equity market component of our final ratio is built upon the most reliable and consistent historical data available.

Part II: Navigating the Complexities of Historical Gold Price Data

The primary challenge outlined in the user query is the difficulty in obtaining long-term historical gold price data through standard, modern Python APIs. This section directly confronts this issue, first by diagnosing the root cause of the problem within the widely-used FRED database and then by presenting a robust, multi-stage solution for acquiring the necessary data from alternative sources.

2.1 The FRED Conundrum: Uncovering the Discontinued Benchmark Series

Initial attempts to use the fredapi library or search the FRED website for historical gold prices often fail to yield data prior to the late 1990s or early 2000s. This is not a user error but a consequence of a significant change in data availability on the FRED platform.

For many years, the global benchmark for the daily price of gold was the London Bullion Market Association (LBMA) Gold Fix. FRED provided this data through two key series: GOLDAMGBD228NLBM for the morning fix and GOLDPMGBD228NLBM for the afternoon fix.[5] These series were widely used in academic research and financial analysis and provided a consistent daily price history extending back several decades.[5]

However, as confirmed by FRED's own blog posts and the disappearance of these series from academic citations in recent years, these key datasets have been removed from the FRED database.[9] This discontinuation is the central reason why simple API calls for long-term gold prices now fail. The removal of a primary, publicly-available data series by a major institution like the St. Louis Fed created a significant data gap for researchers and analysts, forcing a reliance on alternative methods.

With the benchmark series gone, a search for "gold price" on FRED now returns a variety of other related, but fundamentally different, datasets. These include:

- **Producer Price Indexes (PPI):** Series like "Producer Price Index by Industry: Gold Ore Mining" measure the average change over time in the selling prices received by domestic producers for their output.[11] This price is for raw or semi-processed ore and is influenced by mining costs, extraction technology, and labor, making it an unsuitable proxy for the price of refined bullion traded on the international market.
- **Import/Export Price Indexes:** These series track the price of nonmonetary gold as part of international trade data.[13] While related to the global gold price, they reflect trade-specific valuations and are not a direct measure of the spot market price.
- **Financial Derivatives and Volatility Indexes:** FRED also lists data for instruments like

the CBOE Gold ETF Volatility Index (GVZ) or various NASDAQ gold mining indexes.[14] These are derivative products that measure market expectations or the performance of mining company stocks, not the price of the physical commodity itself.

Using any of these available FRED series as a substitute for the historical spot price of gold would introduce significant analytical errors. The distinction between a direct commodity price and a related economic indicator is paramount. The abundance of these proxy series can be misleading, underscoring the importance of carefully evaluating the definition and economic meaning of any dataset before using it in an analysis.

2.2 Primary Solution - The Data Aggregator Strategy with Macrotrends

Given the void left by the discontinued FRED series, the most effective strategy for obtaining free, long-term historical gold price data is to turn to a reputable third-party data aggregator. For this purpose, macrotrends.net stands out as a primary candidate.

Macrotrends provides clean, well-structured historical data tables for a wide range of economic and financial indicators, including a monthly gold price series that extends back to the early 1970s, covering the entire modern floating-price era.[18] The site's data is of sufficient quality to be cited in academic research, lending it a degree of credibility that is essential for this analysis.[20]

Since Macrotrends does not offer a public API, the data must be accessed programmatically through web scraping. The following Python code demonstrates a robust method for this task using the requests and BeautifulSoup libraries to parse the HTML of the webpage and pandas to structure the extracted data.

This script:

1. Fetches the HTML content of the Macrotrends historical gold prices page.
2. Uses BeautifulSoup to parse the HTML and locate the data table. It specifically looks for the script tag containing the raw data, a common practice on modern data-centric websites.
3. Extracts and cleans the raw data string, preparing it for conversion into a structured format.
4. Processes the cleaned data into a pandas DataFrame with Date and Gold_Price_USD columns.
5. Performs necessary data type conversions and sets the Date as the index.

Python

```python
import re
import json

def get_macrotrends_gold_data():
    """
    Scrapes historical monthly gold prices from macrotrends.net.

    Returns:
        pandas.DataFrame: A cleaned DataFrame with monthly gold prices from
                          1970 to the present.
    """
    url = "https://www.macrotrends.net/1333/historical-gold-prices-100-year-chart"
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/58.0.3029.110 Safari/537.36'}

    try:
        response = requests.get(url, headers=headers)
        response.raise_for_status()
    except requests.exceptions.RequestException as e:
        print(f"Failed to fetch data from Macrotrends: {e}")
        return None

    soup = BeautifulSoup(response.text, 'html.parser')

    # The data is embedded in a script tag within the HTML
    scripts = soup.find_all('script')
    data_script = None
    for script in scripts:
        if 'var data = ' in str(script):
            data_script = str(script)
            break

    if not data_script:
        print("Could not find the data script on the page.")
        return None

    # Use regex to find the JSON-like data structure within the script
    json_str_match = re.search(r'var data = (\[.*?\]);', data_script)
    if not json_str_match:
        print("Could not extract JSON data from the script.")
        return None

    json_data = json.loads(json_str_match.group(1))

    # Convert the list of dictionaries to a DataFrame
    df = pd.DataFrame(json_data)

    if 'date' not in df.columns or 'v1' not in df.columns:
        print("The expected columns ('date', 'v1') were not found in the scraped data.")
        return None

    df.rename(columns={'date': 'Date', 'v1': 'Gold_Price_USD'}, inplace=True)

    # Convert date to datetime and price to numeric
    df = pd.to_datetime(df)
    df = pd.to_numeric(df, errors='coerce')

    df.set_index('Date', inplace=True)
    df.sort_index(inplace=True)

    print("Macrotrends gold price data successfully scraped and processed:")
    print(df.head())
    print(df.tail())

    return df

# Execute the function
df_gold_modern = get_macrotrends_gold_data()
```

This approach successfully retrieves the necessary data for the modern era, overcoming the limitations of official data portals.

2.3 Evaluation of Alternative Free Sources

To ensure an exhaustive investigation, it is prudent to evaluate other potential sources for long-term gold price data. While several platforms offer historical prices, they vary in depth, accessibility, and reliability.

- **Trading Economics:** This platform claims to have daily gold price data reaching back to 1968.[21] While this offers greater frequency, the data is often presented within interactive charts that can be more complex to scrape reliably. For a monthly analysis, the additional granularity is not required and may introduce unnecessary complexity.
- **GoldPrice.org:** This site provides various historical charts and data points, including 30-year and 36-year histories.[22] However, like Trading Economics, the data is primarily designed for interactive viewing rather than programmatic download, and the total historical depth may not exceed that of Macrotrends.
- **World Gold Council:** As an official industry body, the World Gold Council offers authoritative data, with some monthly series going back to 1978.[23] While reliable, this is a shorter history than what is available from aggregators and may require navigating a more complex data portal.

A systematic comparison of these sources reveals that for the specific goal of obtaining a free, monthly price series extending to the beginning of the floating-price era, Macrotrends offers the best combination of historical depth and relative ease of programmatic access via scraping.

Source	Data Start Date (Approx.)	Frequency	Access Method	Limitations / Cost
FRED (Current Series)	Varies (e.g., 1984 for price indexes)	Monthly / Daily	fredapi Python library	Does not contain a direct spot price series for gold; only proxies like PPIs.
Macrotrends	~1970 (monthly)	Monthly, Daily	Web Scraping	Free. Relies on stable website structure; no official API.
Trading Economics	~1968	Daily	Web Scraping	Free. Data can be more difficult to extract from charting components.

World Gold Council	1978	Monthly, Quarterly, Annually	Data Hub (Manual Download)	Free. Shorter history than aggregators.
LBMA / ICE (Official)	1968 (for USD)	Daily	Licensed Data Feed	Requires a commercial license from ICE Benchmark Administration (IBA); not free.

2.4 Contextual Aside: The Official (but Restricted) LBMA Benchmark

It is important to understand the official source of the global gold price to appreciate why free access is limited. The London Bullion Market Association (LBMA) oversees the benchmark, which is administered by the ICE Benchmark Administration (IBA).[24] This benchmark, known as the LBMA Gold Price, is set twice daily via an electronic, auditable auction.[7]

Crucially, any party wishing to use this official data for valuation, pricing, or in financial products must obtain a commercial usage license from IBA.[24] This licensing requirement is the fundamental reason why the raw, official historical data is not freely available through public APIs like FRED's anymore. This reality of the financial data landscape necessitates the use of strategies like scraping from reputable aggregators, which collect and republish this data, adhering to the "free" constraint of the user's query.

Part III: Constructing a Continuous Gold Price Series Since 1900

Acquiring the modern, floating-price data is only part of the solution. To create a continuous series extending back to the early 20th century, one must recognize that the fundamental mechanism for gold pricing was different in earlier eras. A single, downloadable dataset covering this entire span does not exist because the market structure has changed.

Therefore, a historically informed, synthetic data series must be constructed.

3.1 A Multi-Era Approach to Data Synthesis

The history of the U.S. dollar's relationship with gold in the 20th century can be divided into three distinct regimes. Understanding these periods is essential for accurately constructing the full time series.[26]

Era	Date Range	Official USD Price per Troy Ounce
The Gold Standard Era	1900 – Jan 1934	$20.67
The Bretton Woods Era	Feb 1934 – Aug 1971	$35.00
The Modern Floating-Price Era	Aug 1971 – Present	Market-Determined

1. **The Gold Standard Era (pre-1934):** From the Coinage Act of 1834 until 1934, the U.S. dollar was defined by a fixed weight of gold, establishing a stable price of approximately $20.67 per troy ounce.[26] For the purposes of international and official transactions, this was the price of gold.
2. **The Bretton Woods Era (1934-1971):** The Gold Reserve Act of 1934 devalued the dollar, re-fixing the official price of gold at $35.00 per troy ounce.[26] This price was the anchor of the Bretton Woods international monetary system, where other currencies were pegged to the dollar, and the dollar itself was convertible to gold by foreign central banks at this rate.
3. **The Modern Floating-Price Era (post-1971):** In August 1971, the "Nixon Shock" unilaterally cancelled the direct convertibility of the U.S. dollar to gold. This act dismantled the Bretton Woods system and allowed the price of gold to "float," meaning its value would henceforth be determined by supply and demand in global markets.[26]

This historical context makes it clear that one cannot simply "find" monthly market prices for gold in 1950 in the same way one can for 2010. The official fixed price was the relevant price for the monetary system. Therefore, the only accurate way to build a continuous series is to programmatically generate the data for the fixed-price eras based on these historical facts and then append the modern market data.

3.2 Python Implementation: Building the Master Gold Time Series

The following Python code implements this multi-era synthesis. It constructs the historical fixed-price data and then merges it with the floating-price data scraped from Macrotrends.

The script executes three main steps:

1. **Generate Fixed-Price Data:** It creates a monthly date range from 1900 to August 1971. It then uses a function to assign the price of $20.67 to dates before February 1934 and $35.00 to dates from February 1934 onwards.
2. **Load Modern Data:** It utilizes the df_gold_modern DataFrame created in Part II, which contains the market-determined prices from the early 1970s to the present.
3. **Concatenate and Finalize:** It combines the historical and modern DataFrames using pd.concat. This creates a single, unified time series representing the price of gold from 1900 to the present day. The resulting DataFrame is resampled to a monthly frequency (MS - month start) to ensure perfect alignment with the Shiller S&P 500 data.

Python

```
def construct_full_gold_series(df_gold_modern):
    """
    Constructs a continuous monthly gold price series from 1900 to present.

    Args:
        df_gold_modern (pandas.DataFrame): DataFrame of modern (post-1970) gold prices.

    Returns:
        pandas.DataFrame: A unified DataFrame with monthly gold prices since 1900.
    """
    if df_gold_modern is None:
        print("Modern gold data is not available. Cannot construct full series.")
        return None

    # Step 1: Generate data for the fixed-price eras
    historical_dates = pd.date_range(start='1900-01-01', end='1971-08-01', freq='MS')
    historical_prices =

    for date in historical_dates:
        if date < pd.to_datetime('1934-02-01'):
            historical_prices.append(20.67)  # Gold Standard price
        else:
            historical_prices.append(35.00)  # Bretton Woods price

    df_gold_historical = pd.DataFrame(
        data={'Gold_Price_USD': historical_prices},
        index=historical_dates
    )
    df_gold_historical.index.name = 'Date'

    # Step 2: Combine with the modern floating-price data
    # Ensure the modern data doesn't overlap incorrectly with the fixed-price era
    df_gold_modern_filtered = df_gold_modern[df_gold_modern.index >
df_gold_historical.index[-1]]

    df_gold_full = pd.concat([df_gold_historical, df_gold_modern_filtered])

    # Resample to month-start frequency to ensure alignment with Shiller data
    df_gold_full = df_gold_full.resample('MS').first().ffill()

    print("Full historical gold price series constructed:")
    print(df_gold_full.loc['1933-12-01':'1934-03-01']) # Show the transition
    print(df_gold_full.loc['1971-07-01':'1971-10-01']) # Show the transition
    print(df_gold_full.tail())

    return df_gold_full

# Execute the function
df_gold_full = construct_full_gold_series(df_gold_modern)
```

This process of data synthesis is a crucial analytical step. It transforms fragmented historical information into a single, continuous, and machine-readable dataset, enabling the final stage of the analysis.

Part IV: Synthesis and Visualization: The S&P 500 Priced in Gold

With both the long-term S&P 500 and the continuous gold price series prepared, the final step is to merge them, calculate the desired ratio, and create a meaningful visualization that reveals the century-long dynamics between financial assets and hard currency.

4.1 Data Unification and Ratio Calculation

The two master DataFrames—df_sp500 from the Shiller dataset and df_gold_full from our synthetic construction—are now ready to be combined. Both are indexed by month, ensuring a straightforward merge.

The following Python code performs this unification:

1. It uses pandas.merge to join the two DataFrames on their date indexes. An inner join ensures that only dates present in both datasets are included.
2. It calculates the core ratio: the price of the S&P 500 index divided by the price of a troy ounce of gold. This new column, SP500_in_Gold, represents the number of gold ounces required to purchase one "unit" of the S&P 500 index.
3. The resulting DataFrame contains all the necessary information for plotting.

Python

```python
def calculate_sp500_in_gold_ratio(df_sp500, df_gold_full):
    """
    Merges S&P 500 and gold data and calculates the ratio.

    Args:
        df_sp500 (pandas.DataFrame): The processed Shiller S&P 500 data.
        df_gold_full (pandas.DataFrame): The constructed full gold price series.

    Returns:
        pandas.DataFrame: A merged DataFrame containing the S&P 500/Gold ratio.
    """
    if df_sp500 is None or df_gold_full is None:
        print("Input data is missing. Cannot calculate ratio.")
        return None

    # Merge the two dataframes on their date index
    df_merged = df_sp500.merge(df_gold_full, left_index=True, right_index=True, how='inner')
    # Calculate the ratio
    df_merged = df_merged / df_merged

    print("S&P 500 in Gold ratio calculated:")
    print(df_merged.head())
    print(df_merged.tail())

    return df_merged

# Execute the function
df_ratio = calculate_sp500_in_gold_ratio(df_sp500, df_gold_full)
```

4.2 Charting a Century of Market Dynamics

A simple line plot is insufficient to convey the rich story contained within a century of financial data. Effective visualization requires specific techniques to handle the vast scale changes and to provide historical context.

The most critical technique for this type of chart is the use of a **logarithmic scale** on the y-axis. Financial time series that exhibit exponential growth over long periods are distorted on a linear scale; early fluctuations appear flat and insignificant, while later movements are exaggerated. A logarithmic scale plots the data based on percentage changes, making movements of the same relative magnitude appear visually consistent across the entire timeline. For example, a 50% drop in the ratio in the 1930s will have the same vertical size as a 50% drop in the 2000s, allowing for accurate comparison of market cycles.

Furthermore, annotating the chart with key historical events transforms it from a data plot into an analytical narrative. By marking major market peaks, troughs, and policy shifts, the viewer can immediately connect the abstract movements of the ratio to their real-world drivers.

The following code uses matplotlib to generate a professional, presentation-quality chart incorporating these best practices.

Python

```python
def plot_sp500_in_gold(df_ratio):
    """
    Creates a historical plot of the S&P 500 priced in gold ounces.

    Args:
        df_ratio (pandas.DataFrame): The DataFrame containing the calculated ratio.
    """
    if df_ratio is None:
        print("Ratio data is not available for plotting.")
        return

    plt.style.use('seaborn-v0_8-whitegrid')
    fig, ax = plt.subplots(figsize=(15, 8))

    # Plot the ratio
    ax.plot(df_ratio.index, df_ratio, color='navy', label='S&P 500 / Gold Ratio')

    # --- CRITICAL: Use a logarithmic scale for the y-axis ---
    ax.set_yscale('log')

    # Formatting the chart
    ax.set_title('S&P 500 Priced in Ounces of Gold (1900-Present)', fontsize=18, pad=20)
    ax.set_ylabel('Ounces of Gold to Buy the S&P 500 Index (Log Scale)', fontsize=12)
    ax.set_xlabel('Year', fontsize=12)

    # Add annotations for key historical events
    annotations = {
        '1929-09-01': '1929 Peak',
        '1980-01-01': '1980 Trough\n(High Inflation)',
        '2000-08-01': '2000 Peak\n(Dot-com Bubble)',
        '1971-08-15': 'End of Gold\nConvertibility',
        '2011-08-01': '2011 Trough\n(Post-GFC)',
    }

    for date_str, label in annotations.items():
        date = pd.to_datetime(date_str)
        if date in df_ratio.index:
            y_val = df_ratio.loc
            ax.text(date, y_val, label, ha='center', va='bottom', fontsize=9, alpha=0.8,
                    bbox=dict(boxstyle="round,pad=0.3", fc="yellow", ec="black", lw=0.5, alpha=0.5))
            ax.axvline(x=date, color='r', linestyle='--', linewidth=0.5, alpha=0.7)

    # Improve y-axis tick labels for log scale
    from matplotlib.ticker import ScalarFormatter
    ax.yaxis.set_major_formatter(ScalarFormatter())
    ax.grid(True, which='both', linestyle='--', linewidth=0.5)

    plt.legend()
    plt.tight_layout()
    plt.show()

# Execute the function
if df_ratio is not None:
    plot_sp500_in_gold(df_ratio)
```

4.3 Preliminary Insights from the Data

The resulting chart provides a powerful visual history of investor sentiment. It can be interpreted as a long-term barometer of risk appetite, measuring the cyclical preference for "paper" financial assets (stocks) versus "real" hard assets (gold).

- **Peaks in the Ratio (e.g., 1929, 1960s, 2000):** These periods represent times of extreme optimism in financial assets and the future earnings of corporations. The market is willing to pay a high premium for stocks relative to the perceived safety of gold. These peaks have historically preceded major bear markets or long periods of stock market stagnation in real terms.
- **Troughs in the Ratio (e.g., 1932, 1980, 2011):** These points represent periods of extreme pessimism, where investors flee from financial assets to the perceived safety of hard assets. The 1980 trough, for example, occurred at the height of an inflationary crisis when confidence in paper currency and financial instruments was at a low point. These troughs have often marked generational buying opportunities for equities.

The cyclical nature of the chart is evident. It shows the long-term ebb and flow between fear and greed, illustrating how the relative valuation of stocks and gold has moved through distinct, multi-decade cycles. This visualization, produced through a rigorous and reproducible data workflow, provides the user with a powerful tool for understanding deep market history.

Conclusion: A Robust and Maintainable Data Workflow

This report has detailed a comprehensive, multi-stage methodology for sourcing, constructing, and visualizing over a century of S&P 500 and gold price data using modern, free Python libraries. The analysis successfully overcomes the primary challenge of data acquisition by addressing the discontinuation of key historical series on official platforms and implementing a robust alternative strategy.

The recommended workflow is built on two pillars:

1. **For S&P 500 Data:** Utilizing the academically authoritative Shiller dataset, which provides a consistent, high-quality monthly price series extending back to 1871. This prioritizes data provenance and reliability for the foundational component of the analysis.
2. **For Gold Price Data:** Creating a synthetic, continuous series by combining programmatically generated data from the fixed-price Gold Standard and Bretton Woods eras with modern, floating-price data acquired via web scraping from a reputable aggregator like Macrotrends.

This combined approach yields a complete and historically accurate dataset that enables the calculation and visualization of the S&P 500/Gold ratio from 1900 to the present. The final visualization, enhanced with a logarithmic scale and historical annotations, transforms the raw data into a powerful analytical tool for understanding long-term market cycles and investor sentiment.

For future maintenance and updates, the Python scripts provided are designed to be re-runnable. To update the chart with the latest data, the user would simply need to execute the code again. The script will fetch the latest version of the Shiller data and scrape the most recent prices from Macrotrends. However, it is important to acknowledge a key dependency: the web scraping component relies on the continued stability of the Macrotrends website's HTML structure. Should the website undergo a significant redesign, the scraping portion of the code may require minor adjustments to adapt to the new structure. This is an inherent trade-off when relying on non-API data sources, but it is a manageable one that provides access to invaluable historical data that is otherwise unavailable for free. This robust and transparent workflow provides a durable solution for long-term financial market analysis.

Obras citadas

1. Online Data - Robert Shiller, fecha de acceso: octubre 17, 2025, http://www.econ.yale.edu/~shiller/data.htm
2. S&P 500 Cyclically Adjusted PE Ratio | US Stock Market | Collection - MacroMicro, fecha de acceso: octubre 17, 2025, https://en.macromicro.me/collections/34/us-stock-relative/410/us-sp500-cyclically-adjusted-price-earnings-ratio
3. S&P 500 (SP500) | FRED | St. Louis Fed, fecha de acceso: octubre 17, 2025, https://fred.stlouisfed.org/series/SP500
4. S&P 500 Shiller CAPE Ratio (Monthly) - United States - Hist... - YCharts, fecha de acceso: octubre 17, 2025, https://ycharts.com/indicators/cyclically_adjusted_pe_ratio

5. Exchange Rates, Interest Rates, and the Risk Premium, fecha de acceso: octubre 17, 2025, https://www.ssc.wisc.edu/~cengel/PublishedPapers/AERExchangeRates.pdf
6. Building Probabilistic Causal Models using Collective Intelligence - Squarespace, fecha de acceso: octubre 17, 2025, https://static1.squarespace.com/static/6310761b71fe69088fff310e/t/661e933da87f4d395c7f0a68/1713279808158/Building-Probabilistic-Causal-Models-using-Collective-Intelligence-revised.pdf
7. London Fix Price Chart - GoldSilver, fecha de acceso: octubre 17, 2025, https://goldsilver.com/price-charts/historical-london-fix/
8. 2021 Report on Availability, Quality, and Pricing of Certain Financial Services and Consumer Loan Products, fecha de acceso: octubre 17, 2025, https://occc.texas.gov/sites/default/files/uploads/reports/2021_study_consumer_loan_products.pdf
9. GOLDPMGBD229NLBM | FRED Blog - Federal Reserve Bank of St. Louis, fecha de acceso: octubre 17, 2025, https://fredblog.stlouisfed.org/tag/goldpmgbd229nlbm/
10. GOLDAMGBD228NLBM - FRED Blog - Federal Reserve Bank of St. Louis, fecha de acceso: octubre 17, 2025, https://fredblog.stlouisfed.org/tag/goldamgbd228nlbm/
11. PPI, Gold, BLS - Economic Data Series | FRED | St. Louis Fed, fecha de acceso: octubre 17, 2025, https://fred.stlouisfed.org/tags/series?t=bls%3Bgold%3Bppi
12. Gold - Search Results | FRED | St. Louis Fed, fecha de acceso: octubre 17, 2025, https://fred.stlouisfed.org/searchresults/?st=gold&isTst=1
13. Indexes, Gold - Economic Data Series | FRED | St. Louis Fed, fecha de acceso: octubre 17, 2025, https://fred.stlouisfed.org/tags/series?t=gold%3Bindexes
14. Price, Gold - Economic Data Series | FRED | St. Louis Fed, fecha de acceso: octubre 17, 2025, https://fred.stlouisfed.org/tags/series?t=gold%3Bprice
15. Export Price Index (End Use): Nonmonetary Gold (IQ12260) | FRED | St. Louis Fed, fecha de acceso: octubre 17, 2025, https://fred.stlouisfed.org/series/IQ12260
16. Gold - Economic Data Series | FRED | St. Louis Fed, fecha de acceso: octubre 17, 2025, https://fred.stlouisfed.org/tags/series?t=gold
17. Gold, Daily - Economic Data Series | FRED | St. Louis Fed, fecha de acceso: octubre 17, 2025, https://fred.stlouisfed.org/tags/series?t=daily%3Bgold
18. Gold Prices - 100 Year Historical Chart & Data - Macrotrends, fecha de acceso: octubre 17, 2025, https://www.macrotrends.net/datasets/1333/historical-gold-prices-100-year-chart
19. Gold Prices - 100 Year Historical Chart - Macrotrends, fecha de acceso: octubre 17, 2025, https://www.macrotrends.net/1333/historical-gold-prices-100-year-chart
20. Gold price over the last 100 years. Modified from Macrotrends... - ResearchGate, fecha de acceso: octubre 17, 2025, https://www.researchgate.net/figure/Gold-price-over-the-last-100-years-Modified-from-Macrotrends_fig2_358250599
21. Gold - Price - Chart - Historical Data - News - Trading Economics, fecha de acceso: octubre 17, 2025, https://tradingeconomics.com/commodity/gold

22. Gold Price History - Historical Gold Charts and Prices, fecha de acceso: octubre 17, 2025, https://goldprice.org/gold-price-history.html
23. Gold Spot Prices & Market History | World Gold Council, fecha de acceso: octubre 17, 2025, https://www.gold.org/goldhub/data/gold-prices
24. LBMA Precious Metal Prices, fecha de acceso: octubre 17, 2025, https://www.lbma.org.uk/prices-and-data/precious-metal-prices
25. LBMA Gold Price, fecha de acceso: octubre 17, 2025, https://www.lbma.org.uk/prices-and-data/lbma-gold-price
26. Historical Gold Price Trends & 50 Year Visual Timeline | Alloy, fecha de acceso: octubre 17, 2025, https://thealloymarket.com/historical-gold-price/