Advanced Mendelian Randomization Implementation in Python

Introduction

Mendelian Randomization (MR) is a powerful epidemiological technique that leverages genetic variants as instrumental variables to infer causal relationships between modifiable exposures and health outcomes. This method has gained significant traction in recent years due to its ability to mitigate confounding and reverse causation, which are common pitfalls in traditional observational studies. The surge in the availability of genome-wide association study (GWAS) data has further propelled the adoption of MR, making it a cornerstone in modern genetic epidemiology.

Despite the proliferation of MR methodologies and tools, there remains a need for a comprehensive, user-friendly, and robust Python package that can cater to both novice and advanced users. Existing MR libraries, such as TwoSampleMR, MendelianRandomization, and MR-Base, offer a range of functionalities but often fall short in terms of ease of use, integration with other data analysis workflows, and advanced sensitivity analyses.

This report introduces a new Python package for Mendelian Randomization that aims to address these gaps. Our package is designed to be better than existing MR libraries in several key ways:

1. **User-Friendly Interface**: The package features an intuitive API that simplifies the process of conducting MR analyses. Users can easily import GWAS summary statistics, perform MR analyses, and visualize results with minimal code.

2. **Comprehensive Functionality**: It supports a wide array of MR methods, including univariable and multivariable MR, inverse-variance weighted (IVW) method, MR-Egger regression, weighted median, and mode-based estimation. Additionally, it includes advanced methods for detecting and

correcting for pleiotropy, such as MR-PRESSO and the contamination mixture method.

3. **Integration with Popular Data Science Libraries**: The package seamlessly integrates with popular Python libraries such as NumPy, Pandas, and Matplotlib, allowing users to incorporate MR analyses into their broader data analysis workflows.

4. **Enhanced Visualization**: It provides advanced visualization tools for presenting MR results, including forest plots, scatter plots, and leave-one-out sensitivity plots. These visualizations are designed to be publication-ready, ensuring that users can effectively communicate their findings.

5. **Robustness and Efficiency**: The package is optimized for performance, capable of handling large datasets efficiently. It includes rigorous error-checking and validation steps to ensure the reliability of the results.

6. **Extensive Documentation and Support**: Comprehensive documentation, including tutorials, example analyses, and a detailed API reference, is provided to help users get started quickly. Additionally, an active community forum and dedicated support channels are available for troubleshooting and guidance.

By addressing the limitations of existing MR libraries and incorporating advanced features, this new Python package aims to set a new standard for Mendelian Randomization analyses. Whether you are a researcher looking to explore causal relationships in your data or a clinician seeking to inform public health interventions, this package provides the tools you need to conduct robust and insightful MR studies.

Table of Contents

- Introduction to Mendelian Randomization and Its Importance
  - Overview of Mendelian Randomization
  - Key Principles of Mendelian Randomization
  - Importance of Mendelian Randomization
    - Addressing Confounding and Reverse Causation
    - Applications in Public Health and Clinical Research

- Implementing Mendelian Randomization in Python
    - Existing MR Libraries
    - Advantages of a Python-Based MR Package
- Core Components of a Python-Based MR Package
    - Data Input and Harmonization
    - Instrumental Variable Selection
    - Causal Estimation Methods
    - Sensitivity Analyses
- Example Workflow

- Load GWAS summary statistics
- Harmonize data
- Select instrumental variables
- Estimate causal effect using IVW method
- Perform sensitivity analyses

- Conclusion
- Existing Mendelian Randomization Libraries and Their Limitations
    - Overview of Existing Mendelian Randomization Libraries
        - TwoSampleMR
        - MendelianRandomization
        - MR-PRESSO
    - Limitations of Existing MR Libraries
        - Platform Dependency
        - Performance and Scalability
        - Integration with External Databases
        - User Interface and Learning Curve
    - Proposed Python-Based MR Package
        - Cross-Platform Compatibility
        - Enhanced Performance and Scalability
        - Seamless Integration with External Databases
        - Intuitive User Interface and Documentation
    - Key Features of the Proposed MR Package
        - Data Input and Harmonization
        - Instrumental Variable Selection
        - Causal Estimation Methods
        - Sensitivity Analyses
    - Example Workflow

    - Design and Implementation of the Python Package for Mendelian Randomization
        - Data Input and Harmonization
            - Loading GWAS Summary Statistics
            - Harmonizing Data
        - Instrumental Variable Selection
            - Filtering SNPs
        - Causal Estimation Methods
            - Inverse-Variance Weighted (IVW) Method
            - MR-Egger Regression
        - Sensitivity Analyses
            - Leave-One-Out Analysis
            - Cochran's Q Test
        - Enhanced Performance and Scalability
        - Seamless Integration with External Databases
        - Intuitive User Interface and Documentation
        - Example Workflow
- Load GWAS data
- Harmonize data
- Filter SNPs
- Perform IVW method
- Perform sensitivity analyses
- Print results

Introduction to Mendelian Randomization and Its Importance

Overview of Mendelian Randomization

Mendelian Randomization (MR) is an epidemiological method that leverages genetic variants as instrumental variables to infer causal relationships between modifiable exposures and health outcomes. This technique is particularly valuable in overcoming limitations inherent in traditional observational studies, such as confounding and reverse causation. By using genetic variants that are randomly assorted at conception, MR mimics the randomization process of a controlled trial, thus providing more robust causal inferences.

Key Principles of Mendelian Randomization

1. **Instrumental Variables**: Genetic variants used in MR must satisfy three core assumptions:
    - **Relevance**: The genetic variant is associated with the exposure of interest.
    - **Independence**: The genetic variant is independent of confounders that affect both the exposure and the outcome.
    - **Exclusion Restriction**: The genetic variant affects the outcome only through the exposure, not through any other pathway.
2. **Two-Sample MR**: This approach uses summary statistics from genome-wide association studies (GWAS) for both the exposure and the outcome, allowing for the analysis of large datasets without the need for individual-level data. This method has significantly expanded the scope and feasibility of MR studies (Nature).

Importance of Mendelian Randomization

Addressing Confounding and Reverse Causation

Traditional observational studies often struggle with confounding variables and reverse causation, which can obscure true causal relationships. MR addresses these issues by using genetic variants as proxies for the exposure, which are not influenced by confounders or the outcome itself. This makes MR a powerful tool for causal inference in epidemiology (BMJ).

Applications in Public Health and Clinical Research

MR has been instrumental in identifying causal risk factors for various diseases, informing public health policies, and guiding clinical interventions. For example, MR studies have elucidated the causal effects of lipid levels on coronary artery disease, providing evidence for the benefits of lipid-lowering therapies (Nature Communications).

Implementing Mendelian Randomization in Python

Existing MR Libraries

Several R packages, such as `TwoSampleMR` and `MendelianRandomization`, are widely used for conducting MR analyses. These packages offer a range of methods for univariable and multivariable MR, as well as tools for sensitivity analyses to assess the robustness of causal inferences (MR-Base).

Advantages of a Python-Based MR Package

While R packages are well-established, a Python-based MR package could offer several advantages:

1. **Integration with Data Science Ecosystem**: Python is a popular language in data science, with extensive libraries for data manipulation (e.g., Pandas), statistical analysis (e.g., Statsmodels), and machine learning (e.g., Scikit-learn). A Python-based MR package could seamlessly integrate with these tools, facilitating comprehensive data analysis workflows.

2. **Performance and Scalability**: Python's performance can be enhanced using libraries like NumPy for numerical computations and Dask for parallel processing, making it suitable for handling large-scale genetic datasets.

3. **User-Friendly Interface**: Python's syntax is often considered more intuitive and easier to learn than R, which could lower the barrier to entry for researchers new to MR.

Core Components of a Python-Based MR Package

Data Input and Harmonization

The package should provide functions to import and harmonize GWAS summary statistics, ensuring compatibility between exposure and outcome datasets. This includes handling palindromic SNPs and performing clumping to account for linkage disequilibrium (Nature).

```python import pandas as pd
```

def harmonizedata(exposuredf, outcomedf): # Merge datasets on SNP identifiers mergeddf = pd.merge(exposuredf, outcomedf, on='SNP') # Remove palindromic SNPs mergeddf = mergeddf[~mergeddf['SNP'].str.contains('A/T|T/A|C/G|G/C')] return mergeddf ```

Instrumental Variable Selection

Functions to select strong instrumental variables based on their association with the exposure, typically using a threshold for the p-value (e.g., p < 5x10^-8). The package should also calculate the F-statistic to ensure the strength of the instruments (International Journal of Epidemiology).

```
python def select_instruments(df, p_threshold=5e-8): instruments = df[df['p_value_exposure'] < p_threshold] instruments['F_statistic'] = (instruments['beta_exposure']**2) / (instruments['se_exposure']**2) return instruments[instruments['F_statistic'] > 10]
```

Causal Estimation Methods

The package should implement various MR methods, including:

1. **Inverse-Variance Weighted (IVW)**: The primary method for estimating causal effects, which combines the estimates from individual SNPs weighted by their precision (MR-Base).

```python import numpy as np

def ivwmethod(df): weights = 1 / (df['seoutcome']**2) betaivw = np.sum(weights * df['betaoutcome']) / np.sum(weights) seivw = np.sqrt(1 / np.sum(weights)) return betaivw, se_ivw ```

1. **MR-Egger Regression**: A method that accounts for pleiotropy by allowing for an intercept term in the regression of the outcome on the exposure (Statistical Methods in Medical Research).

```python import statsmodels.api as sm

def mregger(df): X = df['betaexposure'] y = df['betaoutcome'] weights = 1 / (df['seoutcome']**2) X = sm.add_constant(X) model = sm.WLS(y, X, weights=weights).fit() return model.params, model.bse ```

1. **Weighted Median**: A robust method that provides consistent estimates even when up to 50% of the instruments are invalid (International Journal of Epidemiology).

```
python def weighted_median(df): sorted_df =
df.sort_values(by='beta_outcome') weights = 1 /
(sorted_df['se_outcome']**2) cumulative_weights =
np.cumsum(weights) median_index = np.where(cumulative_weights >=
0.5 * np.sum(weights))[0][0] return sorted_df.iloc[median_index]
['beta_outcome']
```

Sensitivity Analyses

To ensure the robustness of the causal estimates, the package should include functions for sensitivity analyses, such as the Cochran's Q test for heterogeneity and the MR-PRESSO method for detecting and correcting for pleiotropic outliers (Nature).

```python from scipy.stats import chi2

def cochranqtest(df): qstatistic = np.sum(((df['betaoutcome'] - df['betaoutcome'].mean())**2) / df['seoutcome']**2) pvalue = chi2.sf(qstatistic, df.shape[0] - 1) return qstatistic, pvalue ```

## Example Workflow

An example workflow using the Python-based MR package might look like this:

```python

# Load GWAS summary statistics

exposuredf = pd.readcsv('exposuregwas.csv') outcomedf = pd.readcsv('outcomegwas.csv')

# Harmonize data

harmonizeddf = harmonizedata(exposuredf, outcomedf)

# Select instrumental variables

instruments = selectinstruments(harmonizeddf)

# Estimate causal effect using IVW method

betaivw, seivw = ivw_method(instruments)

# Perform sensitivity analyses

qstatistic, pvalue = cochranqtest(instruments)

print(f"IVW Estimate: {betaivw} (SE: {seivw})") print(f"Cochran's Q Test: Q={qstatistic}, p={pvalue}") ```

### Conclusion

A Python-based MR package would provide a valuable addition to the existing MR toolkit, offering seamless integration with the broader data science ecosystem, enhanced performance, and user-friendly interfaces. By implementing robust causal estimation methods and comprehensive sensitivity analyses, this package could facilitate more accessible and reliable MR studies, ultimately advancing our understanding of causal relationships in epidemiology.

# Existing Mendelian Randomization Libraries and Their Limitations

## Overview of Existing Mendelian Randomization Libraries

Mendelian Randomization (MR) is a powerful method used to infer causality between risk factors and health outcomes using genetic variants as instrumental variables. Several libraries and packages have been developed to facilitate MR analyses, each with its own strengths and limitations. This section provides an overview of some of the most widely used MR libraries and their limitations.

### TwoSampleMR

The `TwoSampleMR` package, developed for R, is one of the most popular tools for conducting MR analyses. It provides a comprehensive suite of functions for data extraction, harmonization, and causal estimation.

- **Strengths**:
  - Extensive documentation and user support.
  - Integration with large-scale genetic databases like PhenoScanner and the GWAS Catalog.
  - Supports a variety of MR methods, including inverse-variance weighted (IVW), MR-Egger, and weighted median.
- **Limitations**:
  - Limited to R, which may not be ideal for users who prefer Python.
  - Performance issues with very large datasets due to memory constraints.
  - Requires a steep learning curve for users unfamiliar with R.

### MendelianRandomization

The `MendelianRandomization` package, also for R, offers a range of methods for MR analyses, including multivariable MR and methods to account for pleiotropy.

- **Strengths**:
  - Robust methods for handling pleiotropy and heterogeneity.
  - Detailed vignettes and examples for users.
- **Limitations**:
  - Similar to `TwoSampleMR`, it is limited to R.
  - Less intuitive interface compared to some other packages.
  - Limited support for integrating with external databases.

### MR-PRESSO

The `MR-PRESSO` (Mendelian Randomization Pleiotropy RESidual Sum and Outlier) method is designed to detect and correct for horizontal pleiotropy in MR analyses.

- **Strengths**:
  - Effective at identifying and correcting for pleiotropic outliers.
  - Can be used in conjunction with other MR methods.
- **Limitations**:
  - Requires R for implementation.
  - Limited to specific types of pleiotropy and may not handle all forms of bias.
  - Requires additional steps for data preparation and integration with other MR tools.

## Limitations of Existing MR Libraries

While the existing MR libraries provide powerful tools for causal inference, they have several limitations that can hinder their usability and effectiveness.

### Platform Dependency

Most MR libraries are developed for R, which can be a barrier for researchers who prefer Python or other programming languages. This platform dependency limits the accessibility and flexibility of these tools.

### Performance and Scalability

Handling large-scale genetic data can be challenging with existing MR libraries. Performance issues, such as memory constraints and slow processing times, are common when working with extensive datasets. This limitation can be particularly problematic for researchers conducting large-scale MR studies.

### Integration with External Databases

While some MR libraries offer integration with external genetic databases, the process can be cumbersome and limited. Seamless integration with databases like PhenoScanner and the GWAS Catalog is essential for efficient data extraction and harmonization.

### User Interface and Learning Curve

The user interface and learning curve of existing MR libraries can be steep, especially for researchers new to MR or those unfamiliar with R. Intuitive interfaces and comprehensive documentation are crucial for enhancing user experience and adoption.

## Proposed Python-Based MR Package

To address the limitations of existing MR libraries, we propose the development of a Python-based MR package that offers improved accessibility, performance, and integration capabilities. The following sections outline the key features and advantages of this proposed package.

### Cross-Platform Compatibility

Developing the MR package in Python ensures cross-platform compatibility, making it accessible to a broader range of researchers. Python's popularity and ease of use can help lower the barrier to entry for MR analyses.

### Enhanced Performance and Scalability

Leveraging Python's robust data processing libraries, such as NumPy and Pandas, can significantly enhance the performance and scalability of the MR package. Efficient handling of large-scale genetic data is crucial for conducting comprehensive MR studies.

### Seamless Integration with External Databases

The proposed MR package will offer seamless integration with external genetic databases, such as PhenoScanner and the GWAS Catalog. Automated data extraction and harmonization processes will streamline the workflow and reduce the time required for data preparation.

### Intuitive User Interface and Documentation

An intuitive user interface and comprehensive documentation are essential for enhancing user experience. The proposed MR package will include detailed tutorials, examples, and vignettes to guide users through the MR analysis process. Interactive Jupyter notebooks can also be provided to facilitate hands-on learning.

## Key Features of the Proposed MR Package

The proposed Python-based MR package will include several key features to address the limitations of existing libraries and enhance the overall user experience.

### Data Input and Harmonization

The package will support various data input formats, including summary statistics from GWAS and individual-level genetic data. Automated data harmonization processes will ensure consistency and accuracy in the analysis.

### Instrumental Variable Selection

Robust methods for selecting instrumental variables (IVs) will be implemented, including techniques to account for linkage disequilibrium (LD) and pleiotropy. The package will also provide tools for visualizing and assessing the strength of IVs.

### Causal Estimation Methods

A comprehensive suite of causal estimation methods will be available, including IVW, MR-Egger, weighted median, and MR-PRESSO. Users will have the flexibility to choose the most appropriate method for their analysis.

### Sensitivity Analyses

The package will include various sensitivity analysis methods to assess the robustness of causal estimates. Techniques such as leave-one-out analysis, MR-Egger intercept test, and MR-PRESSO outlier detection will be supported.

## Example Workflow

The following example workflow demonstrates how the proposed Python-based MR package can be used to conduct a comprehensive MR analysis.

1. **Load GWAS Summary Statistics**: ```python import mrpackage as mr

   exposuredata = mr.loadgwasdata('exposuregwas.csv') outcomedata = mr.loadgwasdata('outcomegwas.csv') ```

2. **Harmonize Data**: `python harmonized_data = mr.harmonize_data(exposure_data, outcome_data)`

3. **Select Instrumental Variables**: `python ivs = mr.select_ivs(harmonized_data, p_threshold=5e-8)`

4. **Estimate Causal Effect Using IVW Method**: `python ivw_result = mr.ivw(harmonized_data, ivs) print(ivw_result)`

5. **Perform Sensitivity Analyses**: ```python mreggerresult = mr.mregger(harmonizeddata, ivs) print(mreggerresult)

   mrpressoresult = mr.mrpresso(harmonizeddata, ivs) print(mrpressoresult) ```

By addressing the limitations of existing MR libraries and leveraging the strengths of Python, the proposed MR package aims to provide a more accessible, efficient, and user-friendly tool for conducting Mendelian Randomization analyses.

# Design and Implementation of the Python Package for Mendelian Randomization

## Data Input and Harmonization

The first step in implementing a Python package for Mendelian Randomization (MR) is to handle data input and harmonization. This involves loading Genome-Wide Association Study (GWAS) summary statistics and ensuring that the data is consistent and comparable across different datasets.

### Loading GWAS Summary Statistics

The package should support various formats of GWAS summary statistics, including CSV, TSV, and JSON. It should also provide functions to download data directly from public repositories such as PhenoScanner and the GWAS Catalog.

```python import pandas as pd

def loadgwasdata(filepath, fileformat='csv'): if fileformat == 'csv': return pd.readcsv(filepath) elif fileformat == 'tsv': return pd.readcsv(filepath, sep='\t') elif fileformat == 'json': return pd.readjson(file_path) else: raise ValueError("Unsupported file format") ```

### Harmonizing Data

Harmonization involves aligning the effect alleles and ensuring that the data from different sources are comparable. This step is crucial to avoid biases in the MR analysis.

```python def harmonizedata(exposuredata, outcomedata): # Align effect alleles exposuredata['effectallele'] = exposuredata['effectallele'].str.upper() outcomedata['effectallele'] = outcomedata['effect_allele'].str.upper()

# Merge datasets on SNP identifiers
merged_data = pd.merge(exposure_data, outcome_data, on='SNP')

# Ensure alignment of effect alleles
aligned_data = merged_data[merged_data['effect_allele_x'] == merged_data['e
```
```

```
return aligned_data
```

Instrumental Variable Selection

Selecting appropriate instrumental variables (IVs) is a critical step in MR analysis. The package should include functions to filter SNPs based on their association with the exposure and to perform clumping to remove correlated SNPs.

Filtering SNPs

SNPs should be filtered based on their p-value and linkage disequilibrium (LD) to ensure they are strong instruments.

```
python def filter_snps(gwas_data, p_value_threshold=5e-8, ld_threshold=0.001): filtered_snps = gwas_data[gwas_data['p_value'] < p_value_threshold] # Perform LD clumping (this is a simplified example) clumped_snps = filtered_snps.drop_duplicates(subset=['chromosome', 'position'], keep='first') return clumped_snps
```

Causal Estimation Methods

The package should implement various causal estimation methods, including the Inverse-Variance Weighted (IVW) method, MR-Egger regression, and the weighted median method.

Inverse-Variance Weighted (IVW) Method

The IVW method is a commonly used approach in MR analysis. It combines the estimates from multiple SNPs, weighting them by the inverse of their variance.

```python import numpy as np

def ivwmethod(harmonizeddata): betax = harmonizeddata['betax'] betay = harmonizeddata['betay'] sex = harmonizeddata['sex'] sey = harmonizeddata['sey']
```

```
weights = 1 / (se_y ** 2)
ivw_estimate = np.sum(weights * beta_y / beta_x) / np.sum(weights)
ivw_se = np.sqrt(1 / np.sum(weights))

return ivw_estimate, ivw_se
```

MR-Egger Regression

MR-Egger regression is used to detect and adjust for pleiotropy, where SNPs affect the outcome through pathways other than the exposure.

```python import statsmodels.api as sm

def mregger(harmonizeddata): X = harmonizeddata['betax'] Y = harmonizeddata['betay'] weights = 1 / (harmonizeddata['sey'] ** 2)

X = sm.add_constant(X)
model = sm.WLS(Y, X, weights=weights).fit()

return model.params, model.bse
```

Sensitivity Analyses

Sensitivity analyses are essential to assess the robustness of the MR results. The package should include methods such as the leave-one-out analysis, Cochran's Q test for heterogeneity, and the MR-PRESSO method for detecting outliers.

Leave-One-Out Analysis

This analysis involves repeating the MR analysis, leaving out one SNP at a time, to check if any single SNP is driving the results.

```python def leave_one_out_analysis(harmonized_data, method): results = [] for i in range(len(harmonized_data)): subset = harmonized_data.drop(i) estimate, se = method(subset) results.append((estimate, se)) return results
```

Cochran's Q Test

Cochran's Q test is used to detect heterogeneity in the SNP effects, which can indicate pleiotropy.

```python def cochranqtest(harmonizeddata): betay = harmonizeddata['betay'] sey = harmonizeddata['se_y']

ivw_estimate, _ = ivw_method(harmonized_data)
q_statistic = np.sum(((beta_y - ivw_estimate) ** 2) / (se_y ** 2))
p_value = 1 - stats.chi2.cdf(q_statistic, df=len(beta_y) - 1)

return q_statistic, p_value

```

Enhanced Performance and Scalability

To ensure the package is performant and scalable, it should leverage efficient data structures and parallel processing where possible. For example, using NumPy arrays and Dask for parallel computing can significantly speed up the analysis.

```python import dask.dataframe as dd

def parallelivwmethod(harmonizeddata): ddf = dd.frompandas(harmonized_data, npartitions=4)

```
def compute_ivw(partition):
    beta_x = partition['beta_x']
    beta_y = partition['beta_y']
    se_x = partition['se_x']
    se_y = partition['se_y']

    weights = 1 / (se_y ** 2)
    ivw_estimate = np.sum(weights * beta_y / beta_x) / np.sum(weights)
    ivw_se = np.sqrt(1 / np.sum(weights))

    return ivw_estimate, ivw_se
```
```

```
results = ddf.map_partitions(compute_ivw).compute()
return results
```

Seamless Integration with External Databases

The package should provide seamless integration with external databases to facilitate data retrieval and ensure up-to-date analyses. This can be achieved by implementing API clients for popular databases.

```python import requests

def fetchgwasdatafromapi(apiurl, params): response = requests.get(apiurl, params=params) if response.statuscode == 200: return response.json() else: response.raisefor_status() ```

## Intuitive User Interface and Documentation

A user-friendly interface and comprehensive documentation are crucial for the adoption of the package. The package should include clear function names, detailed docstrings, and examples of usage.

```python def ivwmethod(harmonizeddata): """ Perform Inverse-Variance Weighted (IVW) method for Mendelian Randomization.

```
Parameters:
harmonized_data (DataFrame): Harmonized data containing beta_x, beta_y, se_

Returns:
tuple: IVW estimate and standard error.
"""
beta_x = harmonized_data['beta_x']
beta_y = harmonized_data['beta_y']
se_x = harmonized_data['se_x']
se_y = harmonized_data['se_y']

weights = 1 / (se_y ** 2)
ivw_estimate = np.sum(weights * beta_y / beta_x) / np.sum(weights)
ivw_se = np.sqrt(1 / np.sum(weights))
```

```
return ivw_estimate, ivw_se
```

Example Workflow

An example workflow demonstrates how to use the package to perform a complete MR analysis, from loading data to interpreting results.

```python
# Load GWAS data

exposuredata = loadgwasdata('exposuredata.csv') outcomedata = loadgwasdata('outcomedata.csv')

# Harmonize data

harmonizeddata = harmonizedata(exposuredata, outcomedata)

# Filter SNPs

filteredsnps = filtersnps(harmonized_data)

# Perform IVW method

ivwestimate, ivwse = ivwmethod(filteredsnps)

# Perform sensitivity analyses

leaveoneoutresults = leaveoneoutanalysis(filteredsnps, ivwmethod) qstatistic, qpvalue = cochranqtest(filteredsnps)
```

Print results

print(f"IVW Estimate: {ivwestimate}, SE: {ivwse}") print(f"Cochran's Q: {qstatistic}, p-value: {qp_value}") ```

By implementing these features, the proposed Python package for Mendelian Randomization will offer enhanced performance, scalability, and usability compared to existing MR libraries. This will facilitate more robust and efficient causal inference in epidemiological research.

References

- https://augmentationlab.medium.com/writing-a-simulation-i-conways-game-of-life-86202a003040
- https://pypi.org/project/genal-python/
- https://github.com/OpenOmics/mr-seek
- https://github.com/topics/mendelianrandomization
- https://link.springer.com/chapter/10.1007/978-3-031-48465-0_5
- https://noweyr.github.io/
- https://www.publichealth.columbia.edu/academics/non-degree-special-programs/professional-non-degree-programs/skills-health-research-professionals-sharp-training/mendelian-randomization
- https://onlinelibrary.wiley.com/doi/pdf/10.1002/gepi.22544
- https://conwaylife.com/
- https://academic.oup.com/book/40819/chapter/348789403
- https://www.biorxiv.org/content/10.1101/2022.08.30.505937v1
- https://onlinelibrary.wiley.com/doi/10.1002/brb3.3551
- https://wellcomeopenresearch.org/articles/4-186
- https://conwaylife.com/forums/index.php
- https://conwaylife.com/?rle=9b2o$9bobo$4b2o6bo7b2o$2obo2bo2bo2bo7b2o$2o2b2o6bo$9bobo$9b2o!&name=(p30) queen bee shuttle
- https://www.quantamagazine.org/maths-game-of-life-reveals-long-sought-repeating-patterns-20240118/
- http://py-merp.github.io/intro.html
- https://www.medrxiv.org/content/10.1101/2024.05.23.24307776v1
- https://www.nature.com/articles/s41467-019-14156-4

- https://math.stackexchange.com/questions/1536288/what-are-the-practical-uses-of-game-of-life-or-langtons-ant
- https://www.nature.com/articles/s43586-021-00092-5
- https://www.sciencegate.app/keyword/693600
- https://www.thelancet.com/journals/landia/article/PIIS2213-8587(23)00348-0/fulltext
- https://academic.oup.com/ije/article/51/6/2031/6671803
- https://en.wikipedia.org/wiki/Conway'sGameof_Life
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7614635/
- https://www.medrxiv.org/content/10.1101/2024.05.23.24307776v1.full.pdf
- https://amymariemason.github.io/MR/
- https://medium.com/@alper.bulbul1/exploring-the-causal-pathways-with-python-implementing-mendelian-randomization-methods-in-research-adca766bcf2f
- https://www.nature.com/articles/s43856-024-00530-x
- https://www.nytimes.com/2020/12/28/science/math-conway-game-of-life.html
- https://www.cambridge.org/core/journals/psychological-medicine/article/mendelian-randomization-causal-inference-leveraging-genetic-data/333B418DF7B88B53D58032EA316D0834
- https://bmjopen.bmj.com/content/bmjopen/13/9/e072087.full.pdf?with-ds=yes
- https://biofish.medium.com/pyconway-6dc2479de409
- https://www.discovermagazine.com/the-sciences/mathematicians-prove-the-omniperiodicity-of-conways-game-of-life?ref=gorillasun.de
- https://github.com/anpu9/UCB-CSC1C-proj1-Game-of-Life-
- https://longnow.org/ideas/conways-game-of-life-and-three-millennia-of-human-history/
- https://www.nature.com/articles/s41398-024-02950-8
- https://pubmed.ncbi.nlm.nih.gov/38706291/
- https://github.com/matthijsknigge/mendelianrandomization
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8725623/
- https://conwaylife.com/forums/viewtopic.php?t=6489
- https://conwaylife.com/wiki/Conway'sGameof_Life
- https://www.nature.com/articles/s41598-024-59326-7
- https://github.com/peteryin21/py-merp
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11161388/

- https://www.youtube.com/watch?v=aabO1dwT2Fs
- https://pubmed.ncbi.nlm.nih.gov/38263619/
- https://www.frontiersin.org/articles/10.3389/fcimb.2016.00057/full