

Article

A GWAS data fetcher with AI for Mendelian Randomization analysis

WenJun Zhang

School of Life Sciences, Sun Yat-sen University, Guangzhou, China

E-mail: zhwj@mail.sysu.edu.cn, wjzhang@iaees.org

Received 28 September 2025; Accepted 10 October 2025; Published online 12 October 2025; Published 1 December 2026



Abstract

In present study, a GWAS (Genome-Wide Association Study) data fetcher with AI was developed. It is a web-based tool that generates genetic variant data for Mendelian Randomization (MR) analysis using AI language models. It supports several AI services as DeepSeek, Google Gemini, and OpenAI GPT, etc. The fetcher supports both univariate and multivariate MR analyses. In the fetcher, the input are exposure variable(s) and outcome variable, the output are GWAS exposure and outcome data files. By leveraging AI to generate realistic data, users can practice MR analysis without accessing restricted genetic databases, test analysis pipelines safely, learn GWAS data structure and format, experiment with different exposure-outcome combinations, and develop and validate bioinformatics tools.

Keywords GWAS; data fetcher; Artificial Intelligence (AI); Large Model (LM); web-based tool; Mendelian Randomization (MR).

<p>Network Pharmacology ISSN 2415-1084 URL: http://www.iaees.org/publications/journals/np/online-version.asp RSS: http://www.iaees.org/publications/journals/np/rss.xml E-mail: networkpharmacology@iaees.org Editor-in-Chief: WenJun Zhang Publisher: International Academy of Ecology and Environmental Sciences</p>
--

1 Introduction

Genome-Wide Association Study (GWAS) data is a collective term for data reflecting the effects of SNPs on phenotypes. GWAS data provide information on the association between genotype and phenotype. GWAS data can be used to identify SNPs that are significantly associated with specific phenotypes (GWASLab, 2024; Zhang, 2025a, 2026). As a methodology for detecting causality in observational studies, Mendelian randomization (MR) (Burgess et al., 2013; Burgess and Bowden, 2015; Burgess et al., 2016; Hartwig et al., 2017; Zhang, 2025a, b) use GWAS as the major data source.

In an earlier study I developed a web tool for fetching GWAS data from multiple sources (Zhang, 2016). It is a most reliable way to fetch GWAS data. However in some situations it will meet strict restrictions in accessing public databases. Owing to significant advantages, researchers are expected to widely use the data fetched and generated by AI (Extance, 2025). For this reason, in present study I developed a GWAS (Genome-Wide Association Study) data fetcher with AI. It is a web-based tool that generates genetic variant data for Mendelian Randomization (MR) analysis using AI language models. It supports several AI services as

DeepSeek, Google Gemini, and OpenAI GPT, etc. The tool supports both univariate and multivariate MR analyses. Using AI to generate realistic data, users can fetch GWAS data without accessing restricted genetic databases, experiment with different exposure-outcome combinations, and develop and validate bioinformatics tools.

2 Overview, Purpose and System Architecture

2.1 Overview

GWAS Data Fetcher with AI is a web-based tool that generates genetic variant data for Mendelian Randomization (MR) analysis using AI language models. It supports several AI services: DeepSeek, Google Gemini, and OpenAI GPT, etc.

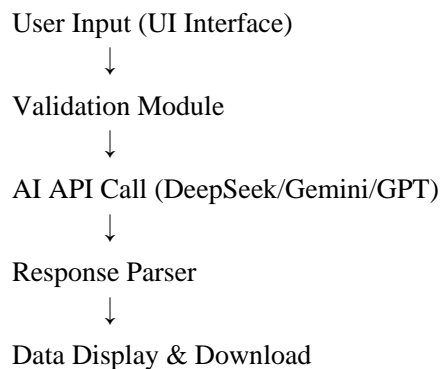
The GWAS Data Generator with AI is a powerful tool that bridges the gap between learning genetic epidemiology concepts and applying them. By leveraging AI to generate realistic data, users can:

- Practice MR analysis without accessing restricted genetic databases
- Test analysis pipelines safely
- Learn GWAS data structure and format
- Experiment with different exposure-outcome combinations
- Develop and validate bioinformatics tools

2.2 Purpose

Generate realistic exposure and outcome CSV files containing genetic variant data (SNPs) for use in GWAS (Genome-Wide Association Studies) and Mendelian Randomization analyses.

2.3 System Architecture



3 Algorithmic Description

3.1 Core Algorithms

A. Analysis Type Handler Algorithm

```

FUNCTION handleAnalysisTypeChange(analysisType):
IF analysisType == "multivariate" THEN
SHOW addExposureButton
ALLOW multiple exposure inputs
ELSE IF analysisType == "univariate" THEN
HIDE addExposureButton
KEEP only first exposure input
REMOVE additional exposure inputs
END IF
  
```

END FUNCTION

B. Input Validation Algorithm

FUNCTION validateInputs(apiKey, exposures, outcome, analysisType):

IF apiKey is empty THEN

RETURN error "API key required"

END IF

IF exposures array is empty THEN

RETURN error "At least one exposure required"

END IF

IF outcome is empty THEN

RETURN error "Outcome variable required"

END IF

IF analysisType == "multivariate" AND exposures.length < 2 THEN

RETURN error "Multivariate requires >= 2 exposures"

END IF

RETURN success

END FUNCTION

C. AI API Call Algorithm

FUNCTION fetchAIData(service, apiKey, exposures, outcome, analysisType):

prompt = constructPrompt(exposures, outcome, analysisType)

SWITCH service:

CASE "deepseek":

endpoint = "https://api.deepseek.com/chat/completions"

payload = {

model: "deepseek-chat",

messages: [systemMessage, userMessage],

temperature: 0.7,

max_tokens: 4000

}

headers = {

"Authorization": "Bearer " + apiKey,

"Content-Type": "application/json"

}

CASE "gemini":

endpoint =

"https://generativelanguage.googleapis.com/v1/models/gemini-1.5-flash:generateContent?key=" + apiKey

payload = {

contents: [{parts: [{text: prompt}]}],

generationConfig: {

```

temperature: 0.7,
maxOutputTokens: 4000
}
}

```

```

CASE "gpt":
endpoint = "https://api.openai.com/v1/chat/completions"
payload = {
model: "gpt-4o-mini",
messages: [systemMessage, userMessage],
temperature: 0.7,
max_tokens: 4000
}
headers = {
"Authorization": "Bearer " + apiKey,
"Content-Type": "application/json"
}
END SWITCH

```

```

response = HTTP_POST(endpoint, headers, payload)
IF response.status != 200 THEN
THROW handleAPIError(response, service)
END IF
RETURN parseResponse(response, service)
END FUNCTION

```

D. Response Parsing Algorithm

```

FUNCTION parseAIResponse(content):
// Extract exposure data
exposurePattern = /EXPOSURE_DATA:([\s\S]*?)(?=OUTCOME_DATA:|$)/
exposureMatch = REGEX_MATCH(content, exposurePattern)

// Extract outcome data
outcomePattern = /OUTCOME_DATA:([\s\S]*?)$/
outcomeMatch = REGEX_MATCH(content, outcomePattern)

IF NOT (exposureMatch AND outcomeMatch) THEN
THROW error "Invalid response format"
END IF

exposureData = cleanCSVData(exposureMatch[1])
outcomeData = cleanCSVData(outcomeMatch[1])

RETURN {exposureData, outcomeData}
END FUNCTION

```

E. CSV Data Cleaning Algorithm

```
FUNCTION cleanCSVData(rawData):  
  // Remove markdown code blocks  
  data = REMOVE_PATTERN(rawData, /``csv/gi)  
  data = REMOVE_PATTERN(data, /``/g)  
  
  // Trim whitespace  
  data = TRIM(data)  
  
  // Normalize line breaks  
  data = REPLACE(data, "\r\n", "\n")  
  data = REPLACE(data, "\r", "\n")  
  
  RETURN data  
END FUNCTION
```

F. Error Handling Algorithm

```
FUNCTION handleAPIError(response, service):  
  statusCode = response.status  
  errorData = PARSE_JSON(response.body)  
  
  SWITCH service:  
  
    CASE "deepseek":  
      IF statusCode == 402 OR errorData.error.code == "insufficient_balance" THEN  
        RETURN "Insufficient balance. Add credits at platform.deepseek.com"  
      ELSE IF statusCode == 401 THEN  
        RETURN "Invalid API key"  
      END IF  
  
    CASE "gemini":  
      IF statusCode == 404 THEN  
        RETURN "Model not found. Check Google AI documentation"  
      ELSE IF statusCode == 400 AND errorData contains "API key" THEN  
        RETURN "Invalid API key"  
      ELSE IF statusCode == 429 THEN  
        RETURN "Rate limit exceeded"  
      END IF  
  
    CASE "gpt":  
      IF statusCode == 401 THEN  
        RETURN "Invalid OpenAI API key"  
      ELSE IF statusCode == 429 THEN  
        RETURN "Rate limit or insufficient quota"  
      ELSE IF statusCode == 400 THEN
```

```

RETURN "Bad request to OpenAI API"
END IF
END SWITCH

```

```

RETURN "API Error: " + statusCode + " " + errorData.message
END FUNCTION

```

G. File Download Algorithm

```

FUNCTION downloadCSV(content, filename):
blob = CREATE_BLOB(content, "text/csv")
url = CREATE_OBJECT_URL(blob)

```

```

anchorElement = CREATE_ELEMENT("a")
anchorElement.href = url
anchorElement.download = filename

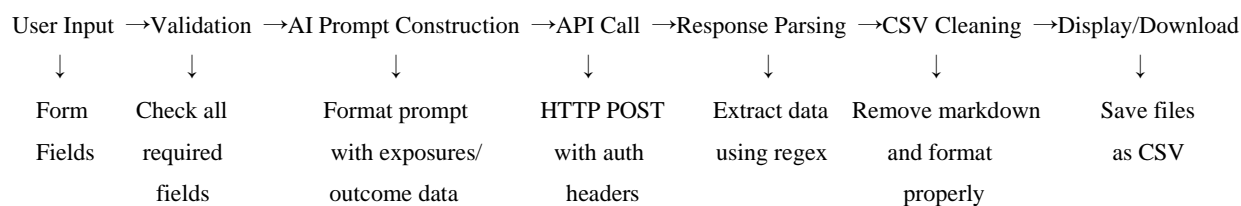
```

```

APPEND_TO_BODY(anchorElement)
CLICK(anchorElement)
REMOVE_FROM_BODY(anchorElement)
REVOKE_OBJECT_URL(url)
END FUNCTION

```

3.2 Data Flow Diagram



4 User Manual

4.1 Getting Started

Prerequisites

- Modern web browser (Chrome, Firefox, Safari, Edge)
- Internet connection
- API key from one of the supported services

Obtaining API Keys

(1) DeepSeek API (Recommended for beginners - Free tier available)

- Visit: <https://platform.deepseek.com/>
- Create account
- Navigate to API Keys section
- Generate new key
- Free credits provided

(2) Google Gemini API

- Visit: <https://makersuite.google.com/app/apikey>
- Sign in with Google account

- Click "Create API Key"
- Copy the generated key
- Free tier available

(3) OpenAI GPT API

- Visit: <https://platform.openai.com/api-keys>
- Create account
- Add payment method
- Generate API key
- Pay-per-use pricing

4.2 Step-by-Step Guide

Step 1: Select AI Service

1. Open the application in your web browser
2. Click the "AI Service" dropdown
3. Select your preferred service:
 - o **DeepSeek**: Good balance of quality and cost
 - o **Gemini**: Google's AI, free tier available
 - o **GPT**: High quality, pay-per-use

Step 2: Enter API Key

1. Paste your API key in the "API Key" field
2. The key is masked for security (shown as dots)
3. Note the service info below the field

Step 3: Choose Analysis Type

Univariate Analysis (Single exposure):

- Select "Univariate" radio button
- Used for testing single exposure → outcome relationship
- Example: BMI → Heart Disease

Multivariate Analysis (Multiple exposures):

- Select "Multivariate" radio button
- Used for testing multiple exposures → outcome relationship
- Example: BMI + Cholesterol + Blood Pressure → Heart Disease

Step 4: Enter Exposure Variables

For Univariate:

1. Enter one exposure trait (e.g., "Body Mass Index")

For Multivariate:

1. Enter first exposure trait
2. Click "+ Add Another Exposure" button
3. Enter additional exposures
4. Click "Remove" button to delete unwanted exposures
5. Minimum 2 exposures required

Examples of Exposure Traits:

- Body Mass Index (BMI)
- HDL Cholesterol
- LDL Cholesterol
- Systolic Blood Pressure

- Triglycerides
- C-Reactive Protein
- Vitamin D levels

Step 5: Enter Outcome Variable

1. Enter the outcome/disease of interest
2. Examples:
 - o Coronary Artery Disease
 - o Type 2 Diabetes
 - o Stroke
 - o Alzheimer's Disease
 - o Cancer

Step 6: Generate Data

1. Click the "Generate Data Files" button
2. Wait for AI processing (typically 10-30 seconds)
3. A loading spinner will appear
4. Do not refresh the page during generation

Step 7: Review Generated Data

The system generates two CSV files:

Exposure File Format:

```
csv
SNP,beta,se,pval,samplesize,effect_allele,other_allele
rs123456,0.05,0.02,1.5e-8,100000,A,G
rs789012,0.03,0.01,3.2e-9,100000,T,C
```

Outcome File Format:

```
csv
SNP,beta,se,pval,samplesize,effect_allele,other_allele
rs123456,0.02,0.01,2.1e-8,150000,A,G
rs789012,0.01,0.01,4.5e-8,150000,T,C
```

Column Definitions:

- SNP:** Genetic variant identifier (rs number)
- beta:** Effect size estimate
- se:** Standard error of the effect
- pval:** Statistical significance (p-value)
- samplesize:** Number of individuals in study
- effect_allele:** Allele associated with trait increase
- other_allele:** Reference/alternative allele

Step 8: Download Files

1. Review the data previews on screen
2. Click "[?] Download Exposure File" to save exposure.csv
3. Click "[?] Download Outcome File" to save outcome.csv
4. Files will be saved to your default downloads folder

Error Messages & Solutions

Error Message Cause Solution

"Please enter your API key" API key field empty Enter valid API key

"Please enter at least one exposure variable" No exposure entered Fill in exposure field

Error Message Cause Solution

"Please enter an outcome variable" No outcome entered Fill in outcome field

"Multivariate analysis requires at least 2 exposure variables"

Only 1 exposure in multivariate mode

Add another exposure or switch to univariate

"Invalid API key" Wrong or expired key

Check API key and generate new one if needed

"Insufficient balance" (DeepSeek) No credits remaining Add credits to your account

"Rate limit exceeded" Too many requests Wait a few minutes and try again

"Invalid response format" AI didn't return proper data Try again or switch AI service

4.3 Best Practices

(1) Choosing Variables

- Use established biomarkers and diseases
- Be specific (e.g., "LDL Cholesterol" vs "Cholesterol")
- Use standard terminology from medical literature

(2) Quality Checks

- Verify SNP IDs start with "rs"
- Check p-values are genome-wide significant ($< 5 \times 10^{-8}$)
- Ensure beta values are reasonable (-0.5 to 0.5)
- Confirm alleles are A, T, C, or G

(3) Cost Management

- Start with DeepSeek (free credits)
- Test with small requests first
- Avoid rapid repeated requests

(4) Data Usage

- Save files immediately after generation
- Keep track of your exposure-outcome combinations
- Document which AI service was used
- Note the date of generation

4.4 Technical Specifications

Supported Data Format

- File Type:** CSV (Comma-Separated Values)
- Encoding:** UTF-8
- Line Endings:** Unix (LF) or Windows (CRLF)
- Number of SNPs:** 15-20 per file

Browser Compatibility

- Chrome 90+
- Firefox 88+
- Safari 14+

- Edge 90+

Network Requirements

- HTTPS connection required
- Outbound connections to:
 - o api.deepseek.com (DeepSeek)
 - o generativelanguage.googleapis.com (Gemini)
 - o api.openai.com (OpenAI)

Security Features

- API keys never stored locally
- Password field masking
- CORS-compliant API calls
- No data transmitted to third parties except chosen AI service

4.5 Use Cases

(1) Educational Purposes

- Learning Mendelian Randomization concepts
- Understanding GWAS data structure
- Practicing genetic epidemiology methods

(2) Research Planning

- Testing analysis pipelines before accessing real data
- Demonstrating methods in grant proposals
- Creating example datasets for publications

(3) Software Development

- Testing MR analysis software
- Validating data processing pipelines
- Creating unit tests

(4) Training & Workshops

- Hands-on genetics workshops
- Bioinformatics training courses
- Statistical genetics tutorials

4.6 Example Workflow

Scenario: Testing obesity's effect on diabetes risk

1. Setup

- o AI Service: DeepSeek
- o Analysis Type: Univariate

2. Input

- o Exposure: "Body Mass Index"
- o Outcome: "Type 2 Diabetes"

3. Generate

- o Click "Generate Data Files"
- o Wait 15 seconds

4. Output

- o Exposure file: 18 SNPs related to BMI
- o Outcome file: Same 18 SNPs in relation to T2D
- o Both files with genome-wide significant associations

5. Next Steps

- Download both files
- Import into MR analysis software (e.g., TwoSampleMR in R)
- Perform MR analysis
- Interpret results

4.7 Troubleshooting

Problem: AI returns incomplete data

- Solution:** Try again or switch to different AI service

Problem: Downloaded file won't open

- Solution:** Ensure you're using CSV-compatible software (Excel, Google Sheets, R, Python)

Problem: Generation takes too long

- Solution:** Check internet connection; if > 60 seconds, refresh page and try again

Problem: "Safety filter" error (Gemini)

- Solution:** Rephrase your exposure/outcome variables using more clinical terminology

4.8 Support & Resources

API Documentation:

- DeepSeek: <https://platform.deepseek.com/docs>
- Gemini: <https://ai.google.dev/docs>
- OpenAI: <https://platform.openai.com/docs>

Mendelian Randomization Resources:

- MR-Base: <https://www.mrbase.org/>
- TwoSampleMR package: <https://mrcieu.github.io/TwoSampleMR/>
- MR tutorial: <https://mrcieu.github.io/mrbase-tutorial/>

4.9 License & Disclaimer

Important: This tool generates synthetic data mainly for educational and testing purposes. For real genetic data, consult:

- UK Biobank
- dbGaP
- European Genome-phenome Archive
- Other established genetic databases

5 Complete Codes

The following are the HTML+JavaScript codes of the GWAS data fetcher with AI

([http://www.iaees.org/publications/journals/np/articles/2026-11\(3-4\)/dataFetchAI.htm](http://www.iaees.org/publications/journals/np/articles/2026-11(3-4)/dataFetchAI.htm); Fig. 1):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Genetic Data Fetcher with AI</title>
  <style>
    * {
      margin: 0;
      padding: 0;
```

```
    box-sizing: border-box;
  }

body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  min-height: 100vh;
  padding: 20px;
}

.container {
  max-width: 900px;
  margin: 0 auto;
  background: white;
  border-radius: 15px;
  padding: 30px;
  box-shadow: 0 10px 40px rgba(0,0,0,0.3);
}

h1 {
  color: #667eea;
  margin-bottom: 10px;
  font-size: 28px;
}

.subtitle {
  color: #666;
  margin-bottom: 30px;
  font-size: 14px;
}

.form-group {
  margin-bottom: 20px;
}

label {
  display: block;
  margin-bottom: 8px;
  color: #333;
  font-weight: 600;
}

select, input[type="text"], input[type="password"] {
  width: 100%;
  padding: 12px;
  border: 2px solid #e0e0e0;
  border-radius: 8px;
  font-size: 14px;
  transition: border-color 0.3s;
}
```

```
select:focus, input:focus {
  outline: none;
  border-color: #667eea;
}

.radio-group {
  display: flex;
  gap: 20px;
  margin-top: 8px;
}

.radio-label {
  display: flex;
  align-items: center;
  gap: 8px;
  cursor: pointer;
  font-weight: normal;
}

input[type="radio"] {
  width: 18px;
  height: 18px;
  cursor: pointer;
}

.exposures-container {
  border: 2px dashed #e0e0e0;
  border-radius: 8px;
  padding: 15px;
  margin-top: 10px;
}

.exposure-input {
  display: flex;
  gap: 10px;
  margin-bottom: 10px;
}

.exposure-input input {
  flex: 1;
}

.btn {
  padding: 12px 24px;
  border: none;
  border-radius: 8px;
  font-size: 14px;
  font-weight: 600;
  cursor: pointer;
}
```

```
    transition: all 0.3s;
  }

.btn-primary {
  background: #667eea;
  color: white;
  width: 100%;
  margin-top: 10px;
}

.btn-primary:hover:not(:disabled) {
  background: #5568d3;
  transform: translateY(-2px);
}

.btn-primary:disabled {
  background: #ccc;
  cursor: not-allowed;
}

.btn-secondary {
  background: #f0f0f0;
  color: #333;
}

.btn-secondary:hover {
  background: #e0e0e0;
}

.btn-danger {
  background: #dc3545;
  color: white;
  padding: 8px 16px;
}

.btn-danger:hover {
  background: #c82333;
}

.btn-success {
  background: #28a745;
  color: white;
}

.btn-success:hover {
  background: #218838;
}

.loading {
  display: none;
}
```

```
    text-align: center;
    margin: 20px 0;
}

.spinner {
    border: 4px solid #f3f3f3;
    border-top: 4px solid #667eea;
    border-radius: 50%;
    width: 40px;
    height: 40px;
    animation: spin 1s linear infinite;
    margin: 0 auto;
}

@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}

.result {
    display: none;
    margin-top: 20px;
    padding: 20px;
    background: #f8f9fa;
    border-radius: 8px;
}

.result h3 {
    color: #667eea;
    margin-bottom: 15px;
}

.file-preview {
    background: white;
    padding: 15px;
    border-radius: 8px;
    margin-bottom: 15px;
    max-height: 200px;
    overflow-y: auto;
    font-family: 'Courier New', monospace;
    font-size: 12px;
    white-space: pre;
    border: 1px solid #dee2e6;
}

.download-buttons {
    display: flex;
    gap: 10px;
}
```

```

.error {
  background: #f8d7da;
  color: #721c24;
  padding: 12px;
  border-radius: 8px;
  margin-top: 10px;
  display: none;
}

.api-info {
  font-size: 12px;
  color: #666;
  margin-top: 5px;
}
</style>
</head>
<body>
  <div class="container">
    <h1>☐ GWAS Data Fetcher with AI</h1>
    <p class="subtitle">Generate exposure and outcome files using AI services</p>
    <div class="api-info"><a href="http://www.iaees.org/publications/journals/np/articles/2026-11(3-4)/1-Zhang-Abstract.asp">Zhang WJ. 2026. A GWAS data fetcher with AI for Mendelian Randomization analysis. Network Pharmacology, 11(3-4): 62-89</a>
    </div>
    <div class="form-group">
      <div class="form-group">
        <label>AI Service</label>
        <select id="aiService">
          <option value="deepseek">DeepSeek</option>
          <option value="gemini">Gemini</option>
          <option value="gpt">GPT</option>
        </select>
      </div>
      <div class="form-group">
        <label>API Key</label>
        <input type="password" id="apiKey" placeholder="Enter your API key">
        <div class="api-info" id="apiInfo">DeepSeek API - Free tier available</div>
      </div>
      <div class="form-group">
        ☐ How to get API keys:<br>
        • DeepSeek: <a href="https://platform.deepseek.com/" target="_blank">platform.deepseek.com</a><br>
        • Gemini: <a href="https://makersuite.google.com/app/apikey" target="_blank">makersuite.google.com/app/apikey</a><br>
        • GPT: <a href="https://platform.openai.com/api-keys" target="_blank">platform.openai.com/api-keys</a><br>
      </div>
    </div>
  </div>

```

```

<div class="form-group">
  <label>Analysis Type</label>
  <div class="radio-group">
    <label class="radio-label">
      <input type="radio" name="analysisType" value="univariate" checked>
      Univariate
    </label>
    <label class="radio-label">
      <input type="radio" name="analysisType" value="multivariate">
      Multivariate
    </label>
  </div>
</div>

<div class="form-group">
  <label>Exposure Variable(s)</label>
  <div class="exposures-container" id="exposuresContainer">
    <div class="exposure-input">
      <input type="text" class="exposure-trait" placeholder="e.g., BMI">
    </div>
    <div>
      <button class="btn btn-secondary" id="addExposure" style="display:none; margin-top: 10px;">+ Add Another
    Exposure</button>
    </div>
  </div>

  <div class="form-group">
    <label>Outcome Variable</label>
    <input type="text" id="outcome" placeholder="e.g., Coronary Artery Disease">
  </div>

  <button class="btn btn-primary" id="generateBtn">Generate Data Files</button>

  <div class="loading" id="loading">
    <div class="spinner"></div>
    <p style="margin-top: 10px; color: #667eea;">Generating data with AI...</p>
  </div>

  <div class="error" id="error"></div>

  <div class="result" id="result">
    <h3>☐ Generated Data Files</h3>

    <div>
      <h4 style="margin-bottom: 10px;">Exposure File:</h4>
      <div class="file-preview" id="exposurePreview"></div>
    </div>

    <div>
      <h4 style="margin-bottom: 10px;">Outcome File:</h4>

```

```

    <div class="file-preview" id="outcomePreview"></div>
</div>

<div class="download-buttons">
    <button class="btn btn-success" id="downloadExposure">↓ Download Exposure File</button>
    <button class="btn btn-success" id="downloadOutcome">↓ Download Outcome File</button>
</div>
</div>
</div>
</div>

<script>
    let exposureData = "";
    let outcomeData = "";

    // Analysis type change handler
    document.querySelectorAll('input[name="analysisType"]').forEach(radio => {
        radio.addEventListener('change', function() {
            const addBtn = document.getElementById('addExposure');
            const container = document.getElementById('exposuresContainer');

            if (this.value === 'multivariate') {
                addBtn.style.display = 'block';
            } else {
                addBtn.style.display = 'none';
                // Keep only first exposure input
                const inputs = container.querySelectorAll('.exposure-input');
                for (let i = 1; i < inputs.length; i++) {
                    inputs[i].remove();
                }
            }
        });
    });

    // Add exposure input
    document.getElementById('addExposure').addEventListener('click', function() {
        const container = document.getElementById('exposuresContainer');
        const div = document.createElement('div');
        div.className = 'exposure-input';
        div.innerHTML = `
            <input type="text" class="exposure-trait" placeholder="e.g., HDL Cholesterol">
            <button class="btn btn-danger" onclick="this.parentElement.remove()">Remove</button>
        `;
        container.appendChild(div);
    });

    // AI Service change handler
    document.getElementById('aiService').addEventListener('change', function() {
        const apiInfo = document.getElementById('apiInfo');
        const infos = {
            'deepseek': 'DeepSeek API - Free tier available',

```

```

        'gemini': 'Gemini API - Free tier available',
        'gpt': 'GPT API - Pay-per-use pricing'
    };
    apiInfo.textContent = infos[this.value];
});

// Generate button handler
document.getElementById('generateBtn').addEventListener('click', async function() {
    const aiService = document.getElementById('aiService').value;
    const apiKey = document.getElementById('apiKey').value.trim();
    const analysisType = document.querySelector('input[name="analysisType"]:checked').value;
    const exposureInputs = document.querySelectorAll('.exposure-trait');
    const exposures = Array.from(exposureInputs).map(input => input.value.trim()).filter(v => v);
    const outcome = document.getElementById('outcome').value.trim();

    // Validation
    if (!apiKey) {
        showError('Please enter your API key');
        return;
    }

    if (exposures.length === 0) {
        showError('Please enter at least one exposure variable');
        return;
    }

    if (!outcome) {
        showError('Please enter an outcome variable');
        return;
    }

    if (analysisType === 'multivariate' && exposures.length < 2) {
        showError('Multivariate analysis requires at least 2 exposure variables');
        return;
    }

    // Show loading
    document.getElementById('loading').style.display = 'block';
    document.getElementById('result').style.display = 'none';
    document.getElementById('error').style.display = 'none';
    this.disabled = true;

    try {
        const data = await fetchAIData(aiService, apiKey, exposures, outcome, analysisType);
        displayResults(data);
    } catch (error) {
        showError(error.message);
    } finally {
        document.getElementById('loading').style.display = 'none';
        this.disabled = false;
    }
});

```

```

    }
  });

  // Fetch data from AI
  async function fetchAIData(service, apiKey, exposures, outcome, analysisType) {
    const prompt = `Generate realistic genetic variant data for a ${analysisType} Mendelian Randomization analysis.

Exposure trait(s): ${exposures.join(', ')}
Outcome trait: ${outcome}

Generate data in the following CSV format:

For exposure file(s):
SNP,beta,se,pval,samplesize,effect_allele,other_allele

For outcome file:
SNP,beta,se,pval,samplesize,effect_allele

Requirements:
1. Use real SNP IDs (rs numbers)
2. Generate 15-20 variants
3. Beta values should be realistic effect sizes (typically -0.5 to 0.5)
4. Standard errors should be realistic (0.01 to 0.1)
5. P-values should be genome-wide significant (< 5e-8)
6. Use real alleles (A, T, C, G)
7. Sample sizes should be realistic (50000-500000)
8. Make sure SNPs match between exposure and outcome files

Return ONLY the CSV data, no explanations. Format:
EXPOSURE_DATA:
[CSV data]

OUTCOME_DATA:
[CSV data]`;

    let response;
    let responseData;

    if (service === 'deepseek') {
      response = await fetch('https://api.deepseek.com/chat/completions', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': `Bearer ${apiKey}`
        },
        body: JSON.stringify({
          model: 'deepseek-chat',
          messages: [
            {
              role: 'system',

```

```

        content: 'You are a genetics data expert. Generate only CSV data without any markdown
formatting or explanations.'
    },
    {
        role: 'user',
        content: prompt
    }
  ],
  temperature: 0.7,
  max_tokens: 4000
))
});

if (!response.ok) {
  responseData = await response.json().catch(() => ({}));

  // Handle specific DeepSeek errors
  if (response.status === 402 || responseData.error?.code === 'insufficient_balance') {
    throw new Error('Insufficient balance in your DeepSeek account. Please add credits at
https://platform.deepseek.com/');
  } else if (response.status === 401) {
    throw new Error('Invalid DeepSeek API key. Please check your API key.');
```

```

if (!response.ok) {
  responseData = await response.json().catch(() => ({}));

  // Handle specific Gemini errors
  if (response.status === 404) {
    throw new Error('Gemini model not found. The API may have been updated. Please check the
Google AI documentation.');
```

} else if (response.status === 400 && responseData.error?.message?.includes('API key')) {
 throw new Error('Invalid Gemini API key. Please check your API key at
<https://makersuite.google.com/app/apikey>');

```

    } else if (response.status === 429) {
      throw new Error('Gemini API rate limit exceeded. Please wait a moment and try again.');
```

} else {
 throw new Error(responseData.error?.message || `Gemini API Error: \${response.status}
\${response.statusText}`);
 }
 }

 responseData = await response.json();
} else if (service === 'gpt') {
 response = await fetch('https://api.openai.com/v1/chat/completions', {
 method: 'POST',
 headers: {
 'Content-Type': 'application/json',
 'Authorization': `Bearer \${apiKey}`
 },
 body: JSON.stringify({
 model: 'gpt-4o-mini',
 messages: [
 {
 role: 'system',
 content: 'You are a genetics data expert. Generate only CSV data without any markdown
formatting or explanations.'
 },
 {
 role: 'user',
 content: prompt
 }
],
 temperature: 0.7,
 max_tokens: 4000
 })
 });
}

if (!response.ok) {
 responseData = await response.json().catch(() => ({}));

 // Handle specific OpenAI errors
 if (response.status === 401) {

```

        throw new Error('Invalid OpenAI API key. Please check your API key at
https://platform.openai.com/api-keys');
    } else if (response.status === 429) {
        throw new Error('OpenAI API rate limit exceeded or insufficient quota. Please check your usage at
https://platform.openai.com/usage');
    } else if (response.status === 400) {
        throw new Error(responseData.error?.message || 'Bad request to OpenAI API. Please try again.');
```

} else {
 throw new Error(responseData.error?.message || `OpenAI API Error: \${response.status}
\${response.statusText}`);
 }
}

responseData = await response.json();
}

let content;

if (service === 'gemini') {
 content = responseData.candidates?.[0]?.content?.parts?.[0]?.text;

 // Check for safety blocks
 if (!content && responseData.candidates?.[0]?.finishReason === 'SAFETY') {
 throw new Error('Content was blocked by Gemini safety filters. Please try rephrasing your request.');

}
} else {
 content = responseData.choices?.[0]?.message?.content;
}

if (!content) {
 throw new Error('No content received from AI service. Please try again.');

}

return parseAIResponse(content);
}

// Parse AI response
function parseAIResponse(content) {
 const exposureMatch = content.match(/EXPOSURE_DATA:([\s\S]*?)(?=OUTCOME_DATA:\$/i);
 const outcomeMatch = content.match(/OUTCOME_DATA:([\s\S]*?)\$/i);

 if (!exposureMatch || !outcomeMatch) {
 throw new Error('Invalid response format from AI. Please try again.');

}

let exposureData = exposureMatch[1].trim();
let outcomeData = outcomeMatch[1].trim();

// Clean up the data
exposureData = cleanCSVData(exposureData);

```
    outcomeData = cleanCSVData(outcomeData);

    return { exposureData, outcomeData };
}

// Clean CSV data
function cleanCSVData(data) {
    // Remove markdown code blocks
    data = data.replace(/```csv/gi, "").replace(/```/g, "");

    // Remove extra whitespace
    data = data.trim();

    // Ensure proper line breaks
    data = data.replace(/\r\n/g, '\n').replace(/\r/g, '\n');

    return data;
}

// Display results
function displayResults(data) {
    exposureData = data.exposureData;
    outcomeData = data.outcomeData;

    document.getElementById('exposurePreview').textContent = exposureData;
    document.getElementById('outcomePreview').textContent = outcomeData;
    document.getElementById('result').style.display = 'block';
}

// Download handlers
document.getElementById('downloadExposure').addEventListener('click', function() {
    downloadCSV(exposureData, 'exposure.csv');
});

document.getElementById('downloadOutcome').addEventListener('click', function() {
    downloadCSV(outcomeData, 'outcome.csv');
});

// Download CSV file
function downloadCSV(content, filename) {
    const blob = new Blob([content], { type: 'text/csv' });
    const url = window.URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = filename;
    document.body.appendChild(a);
    a.click();
    document.body.removeChild(a);
    window.URL.revokeObjectURL(url);
}
```

```
// Show error
function showError(message) {
    const errorDiv = document.getElementById('error');
    errorDiv.textContent = message;
    errorDiv.style.display = 'block';
}
</script>
</body>
</html>
```

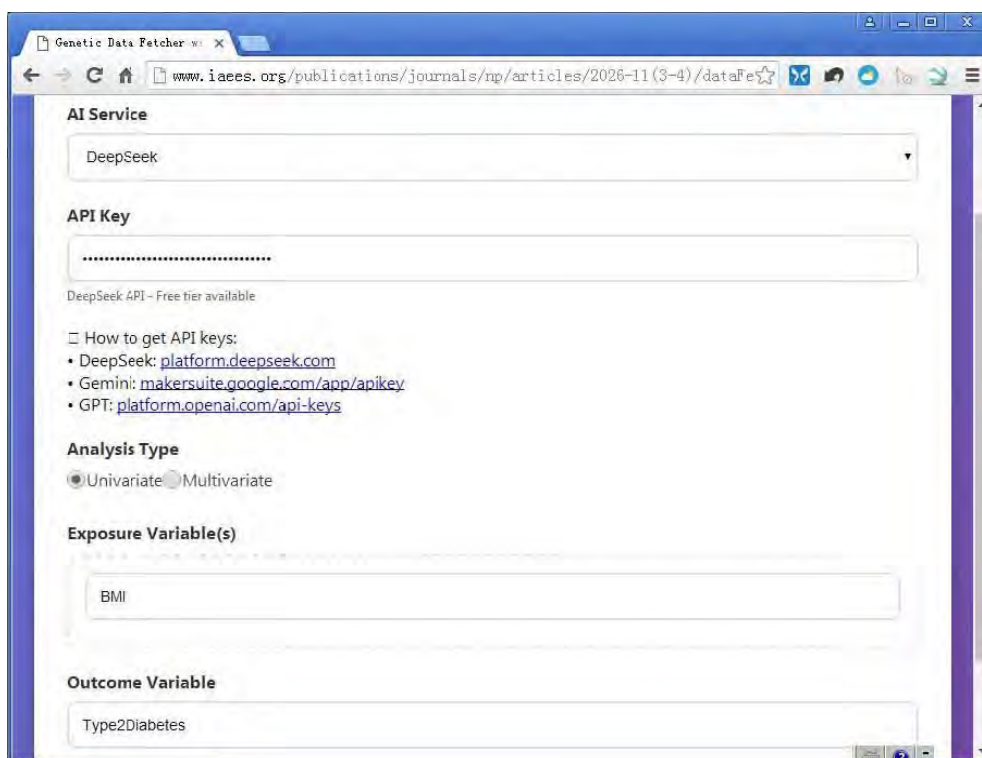
The image shows a web browser window with the URL 'www.iaees.org/publications/journals/np/articles/2026-11(3-4)/dataFe'. The page is titled 'Genetic Data Fetcher' and contains several input fields and options. At the top, there is a dropdown menu for 'AI Service' with 'DeepSeek' selected. Below it is a text input field for 'API Key' containing a series of dots. A note below the API key field says 'DeepSeek API - Free tier available'. There is a section titled 'How to get API keys:' with three bullet points: 'DeepSeek: platform.deepseek.com', 'Gemini: makersuite.google.com/app/apikey', and 'GPT: platform.openai.com/api-keys'. Below this is a section for 'Analysis Type' with two radio buttons: 'Univariate' (selected) and 'Multivariate'. The 'Exposure Variable(s)' section has a text input field containing 'BMI'. The 'Outcome Variable' section has a text input field containing 'Type2Diabetes'.

Fig. 1 GWAS data fetcher with AI for Mendelian Randomization analysis.

6 Demonstration

Use DeepSeek as an example for AI. First, register in <https://platform.deepseek.com> to get the API key and pay for API use (a very little cost). Choose “Univariate” analysis, and exposure variable as “Type2Diabetes”, and outcome variable as “Stroke”. Once pressing “Generate Data Files” and waiting for several seconds, the results as shown in Table 1, will be obtained (Fig. 2).

Table 1 GWAS data fetching results.

	SNP	beta	se	pval	Samplesize	effect_allele	other_allele
Exposure	rs10946398	0.08	0.02	3.20E-09	150000	C	A
	rs4402960	0.12	0.03	1.80E-10	180000	T	G
	rs1801282	0.15	0.04	2.10E-11	200000	C	G
	rs5219	0.09	0.02	4.50E-09	160000	T	C
	rs7903146	0.18	0.05	3.70E-12	220000	T	C
	rs10811661	0.11	0.03	6.20E-10	170000	T	C
	rs8050136	0.13	0.04	8.90E-11	190000	A	C
	rs1111875	0.07	0.02	7.30E-09	140000	C	T
	rs864745	0.14	0.04	4.10E-11	210000	T	C
	rs12779790	0.1	0.03	2.80E-09	155000	G	A
	rs7961581	0.16	0.05	9.40E-12	230000	T	C
	rs4607103	0.08	0.02	5.60E-09	145000	C	T
	rs2943641	0.12	0.03	3.90E-10	175000	T	C
	rs780094	0.09	0.02	8.10E-09	135000	C	T
	rs13266634	0.17	0.05	6.80E-12	240000	C	T
Outcome	rs10946398	0.05	0.01	2.10E-08	250000	C	A
	rs4402960	0.08	0.02	3.50E-09	280000	T	G
	rs1801282	0.06	0.02	4.80E-09	300000	C	G
	rs5219	0.04	0.01	6.30E-08	260000	T	C
	rs7903146	0.09	0.03	1.20E-09	320000	T	C
	rs10811661	0.07	0.02	7.90E-09	270000	T	C
	rs8050136	0.08	0.02	5.40E-09	290000	A	C
	rs1111875	0.03	0.01	9.20E-08	240000	C	T
	rs864745	0.07	0.02	6.10E-09	310000	T	C
	rs12779790	0.05	0.01	3.70E-08	255000	G	A
	rs7961581	0.1	0.03	8.50E-10	330000	T	C
	rs4607103	0.04	0.01	7.40E-08	245000	C	T
	rs2943641	0.06	0.02	4.20E-09	275000	T	C
	rs780094	0.03	0.01	8.90E-08	235000	C	T
	rs13266634	0.11	0.03	2.70E-10	340000	C	T

Generated Data Files

Exposure File:

```
SNP,beta,se,pval,samplesize,effect_allele,other_allele
rs10946398,0.08,0.02,3.2e-09,150000,C,A
rs4402960,0.12,0.03,1.8e-10,180000,T,G
rs1801282,0.15,0.04,2.1e-11,200000,C,G
rs5219,0.09,0.02,4.5e-09,160000,T,C
rs7903146,0.18,0.05,3.7e-12,220000,T,C
rs10811661,0.11,0.03,6.2e-10,170000,T,C
rs8050136,0.13,0.04,8.9e-11,190000,A,C
rs1111875,0.07,0.02,7.3e-09,140000,C,T
rs864745,0.14,0.04,4.1e-11,210000,T,C
rs12779790,0.10,0.03,2.8e-09,155000,G,A
rs7961581,0.16,0.05,9.4e-12,230000,T,C
```

Outcome File:

```
rs5219,0.04,0.01,6.3e-08,260000,T,C
rs7903146,0.09,0.03,1.2e-09,320000,T,C
rs10811661,0.07,0.02,7.9e-09,270000,T,C
rs8050136,0.08,0.02,5.4e-09,290000,A,C
rs1111875,0.03,0.01,9.2e-08,240000,C,T
rs864745,0.07,0.02,6.1e-09,310000,T,C
rs12779790,0.05,0.01,3.7e-08,255000,G,A
rs7961581,0.10,0.03,8.5e-10,330000,T,C
rs4607103,0.04,0.01,7.4e-08,245000,C,T
rs2943641,0.06,0.02,4.2e-09,275000,T,C
rs780094,0.03,0.01,8.9e-08,235000,C,T
rs13266634,0.11,0.03,2.7e-10,340000,C,T
```

Fig. 2 Results of GWAS data fetching.

7 Discussion

AI acts as a powerful intelligent agent that searches, comprehends, and synthesizes already-published and curated GWAS results. The main principle for AI fetching of GWAS data is using natural language processing and knowledge retrieval capabilities to access summaries of GWAS data and then structure that information.

Major procedures and tasks for AI data fetching include:

(1) Data Source Identification and Access

The data sources are publicly available, authoritative databases and scientific literature as GWAS Catalog, PubMed / Europe PMC, published papers, etc.

(2) Natural Language Processing and Information Extraction

When AI accesses a scientific paper or a database entry, it performs Named Entity Recognition (NER) (Identify and extract key biological entities), relationship extraction (Understand the connections between these entities), and statistical data extraction (Parse complex numerical and statistical information).

(3) Data Structuring and Integration

The fetched information is organized into a structured format that is easy for the user to understand and use.

(4) Knowledge Synthesis and Reasoning

AI can go beyond simple retrieval by connecting information from multiple sources, e.g., cross-referencing (Finding all GWAS hits for a specific gene across different studies), functional enrichment (Linking significant SNPs to their potential regulatory roles, e.g., by connecting to eQTL data from GTEx), and summarization (Providing a simple summary of the genetic architecture of a complex trait based on the top findings).

However, AI can only access individual-level data (AI cannot retrieve the raw, individual genotype or phenotype data from controlled-access databases. This requires specific authorizations and approvals), and

perform original GWAS analysis (AI do not run new statistical genetics analyses on raw data). It cannot guarantee 100% completeness (While AI aims for high accuracy, the retrieval is dependent on the source data being up-to-date and correctly parsed).

Therefore, it's always good practice to verify critical findings in the original source for updating fetched data.

References

- Burgess S, Bowden J. 2015. Integrating summarized data from multiple genetic variants in Mendelian randomization: bias and coverage properties of inverse-variance weighted methods. arXiv, 1512.04486.
- Burgess S, Bowden J, Dudbridge F, Thompson SG. 2016. Robust instrumental variable methods using multiple candidate instruments with application to Mendelian randomization. arXiv, 1606.03729.
- Burgess S, Butterworth AS, Thompson SG. 2013. Mendelian randomization analysis with multiple genetic variants using summarized data. *Genetic Epidemiology*, 37: 658-665. doi: 10.1002/gepi.21758
- Extance A. 2025. AI-generated medical data can sidestep usual ethics review, universities say. *Nature News*, 10 Sept 2025. doi: <https://doi.org/10.1038/d41586-025-02911-1>
- GWASLab. 2024. Mendelian Randomization Series No. 1: Basic Concepts Mendelian randomization. <https://gwaslab.org/2021/06/24/mr/>
- Hartwig FP, Smith GD, Bowden J. 2017. Robust inference in summary data Mendelian randomization via the zero modal pleiotropy assumption. *International Journal of Epidemiology*, 46(6): 1985-1998. doi: 10.1093/ije/dyx102
- Zhang WJ. 2025a. A web-based data generator for Mendelian Randomization (MR) analysis. *Network Pharmacology*, 10(3-4): 14-65.
[http://www.iaees.org/publications/journals/np/articles/2025-10\(3-4\)/1-Zhang-Abstract.asp](http://www.iaees.org/publications/journals/np/articles/2025-10(3-4)/1-Zhang-Abstract.asp)
- Zhang WJ. 2025b. Mendelian randomization: Principles and methods. *Network Biology*, 15(2): 24-47.
[http://www.iaees.org/publications/journals/nb/articles/2025-15\(2\)/1-Zhang-Abstract.asp](http://www.iaees.org/publications/journals/nb/articles/2025-15(2)/1-Zhang-Abstract.asp)
- Zhang WJ. 2026. A multi-source GWAS data fetcher for Mendelian Randomization analysis. *Network Pharmacology*, 11(1-2): 1-61.
[http://www.iaees.org/publications/journals/np/articles/2026-11\(1-2\)/1-Zhang-Abstract.asp](http://www.iaees.org/publications/journals/np/articles/2026-11(1-2)/1-Zhang-Abstract.asp)