

Article

A web-based data generator for Mendelian Randomization (MR) analysis

WenJun Zhang

School of Life Sciences, Sun Yat-sen University, Guangzhou 510275, China

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

Received 8 September 2025; Accepted 25 September 2025; Published online 1 October 2025; Published 1 December 2025



Abstract

In present study, a data generator for Mendelian Randomization (MR) analysis was presented. It is a comprehensive web-based tool. The tool takes as input GWAS exposure and outcome data files, and outputs a JSON file for MR analysis. The SNPs correlation matrix is obtained from public databases or by AI (DeepSeek, Gemini, etc.) data construction or as the identity matrix. The tool supports both univariate and multivariate MR analyses. It provides an integrated workflow for processing genetic association data, automatic harmonization, correlation matrix generation, and MR input construction.

Keywords Mendelian Randomization (MR); data generator; web-based tool.

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

1 Introduction

Causal analysis is the fundamental topic in various research areas (Zhang, 2021a-c). Mendelian randomization (MR) is a set of methodology for evaluating causality in observational studies. MR will provide information on causality in situations where randomized controlled trials are not possible. MR is widely used in epidemiological, pharmacological and public health research, especially in evaluating the impact of lifestyle factors, genetic susceptibility and drug targets on disease risk (Zhang, 2015). So far numerous methods for Mendelian Randomization have been developed (Burgess et al., 2013; Burgess and Bowden, 2015; Bowden et al., 2015; del Greco et al., 2015; Burgess et al., 2016; Hartwig et al., 2017; Grant and Burgess, 2020; Lin et al., 2023; Wang, 2023; Xu et al., 2023). Reliable data is the basis for MR analysis. In present study, a web-based data generator for Mendelian Randomization (MR) analysis is proposed. It provides an integrated workflow for processing genetic association data, automatic harmonization, correlation matrix generation, and MR input construction. It supports both univariate and multivariate MR analyses.

2 Data Harmonization

2.1 Data

Suppose there are two data sources, exposure data and outcome data (Table 1), which are from genetic association studies, in the context of Mendelian randomization or meta-analysis.

Table 1 Exposure data and outcome data.

Exposure						
SNP	beta	se	pval	eaf	other_allele	effect_allele
rs9939609	0.045	0.008	1.20E-08	0.42	G	A
rs1801133	0.032	0.007	3.40E-07	0.35	C	T
rs7903146	0.051	0.009	2.10E-09	0.48	G	C
rs1421085	0.038	0.006	8.70E-08	0.52	C	T
rs1558902	0.047	0.008	4.50E-09	0.38	T	A
rs7185735	0.029	0.007	1.90E-06	0.45	C	G
rs71358633	0.041	0.009	6.30E-08	0.41	G	T
rs10821905	0.033	0.006	2.80E-07	0.39	G	C
rs116888810	0.039	0.008	1.10E-08	0.47	T	A
rs12970134	0.036	0.007	5.60E-07	0.44	C	G
rs17782313	0.04	0.007	9.80E-08	0.43	A	G

Outcome					
SNP	beta	se	pval	other_allele	effect_allele
rs9939609	0.156	0.045	0.0006	G	A
rs1801133	0.089	0.038	0.0192	C	T
rs7903146	0.234	0.052	3.80E-05	G	C
rs1421085	0.178	0.041	0.0002	C	T
rs1558902	0.201	0.046	0.0001	T	A
rs7185735	0.112	0.039	0.0041	C	G
rs71358633	0.167	0.05	0.0008	G	T
rs10821905	0.134	0.037	0.0009	G	C
rs116888810	0.189	0.044	0.0001	T	A
rs12970134	0.145	0.04	0.0004	C	G
rs17782313	0.16	0.042	0.0003	A	G

Notes: (1) The outcome data lacks eaf (common if not needed for the analysis). (2) Trailing commas in the outcome data are likely formatting artifacts and can be ignored. (3) All SNPs match between exposure and outcome, enabling paired analysis (e.g., for instrumental variable methods). (4) Data assumes harmonized alleles (effect alleles are consistent across datasets), so no proxy or allele flipping is needed.

The meanings of data columns in the Table 1 are as follows (standard in GWAS summary statistics):

SNP: The identifier for the Single Nucleotide Polymorphism (SNP; genetic variant), e.g., rs9939609. This is the unique label for each locus.

beta: The effect size estimate (regression coefficient) for the association between the SNP and the trait. It represents the change in the outcome (e.g., log-odds for binary traits or mean difference for continuous traits) per allele increase in the effect allele. Positive values indicate an increase; negative indicate a decrease.

se: Standard error of the beta estimate. It quantifies the precision of the beta (smaller se means more precise estimate). Used to compute confidence intervals or p-values.

pval: P-value for the association test (e.g., Wald test), which indicates statistical significance (typically < 0.05 is significant; very small values like 1.20E-08 suggest strong evidence against the null hypothesis of no association).

eaf (Exposure data only): Effect Allele Frequency. The frequency of the effect allele in the population sample (ranges from 0 to 1; e.g., 0.42 means 42% of alleles are the effect allele).

other_allele: The non-effect (reference or alternative) allele at the SNP locus (e.g., G if the effect allele is A).

effect_allele (Exposure and Outcome data): The allele coded as the "risk" or reference allele for the effect size (beta). The beta is estimated per copy of this allele (assuming additive model: 0, 1, or 2 copies). Effect alleles are aligned between exposure and outcome datasets here (e.g., both have A as effect_allele for rs9939609).

2.2 betaX, betaY, betaXse, and betaYse

First, we assume that:

X = Exposure (e.g., the trait or risk factor associated with SNPs in the first dataset, like BMI or cholesterol levels).

Y = Outcome (e.g., the disease or endpoint associated with the same SNPs in the second dataset, like diabetes risk).

We have

betaX (β_X) = Effect of SNP on exposure (beta from exposure data).

betaY (β_Y) = Effect of SNP on outcome (beta from outcome data).

betaXse (β_{Xse}) = Standard error of the effect of SNP on exposure (se from exposure data).

betaYse (β_{Yse}) = Standard error of the effect of SNP on outcome (se from outcome data).

SNPs act as instrumental variables (IVs) to estimate the causal effect of X on Y.

(1) Per-SNP Values

For each SNP i (default in the data generator):

betaX i (β_{X_i}) = Exposure beta for SNP i

betaY i (β_{Y_i}) = Outcome beta for SNP i

betaXse i (β_{Xse_i}) = Exposure se for SNP i

betaYse i (β_{Yse_i}) = Outcome se for SNP i

(2) Aggregated Values (IVW Meta-Analysis Formulas)

If the intent is the aggregated (meta-analyzed) values across all SNPs (e.g., overall SNP-exposure and SNP-outcome effects), use the IVW formulas below. These weight each SNP by its precision ($1/se^2$) and are standard for summarizing multi-SNP effects. If aggregating across all n SNPs (common in MR to get overall instrument strength):

$$\beta_X = \sum_{i=1}^n (w_{X_i} \beta_{X_i}) / \sum_{i=1}^n (w_{X_i}) \quad (\text{aggregated SNP-exposure effect})$$

where $w_{X_i} = 1 / (\beta_{Xse_i})^2$, is the inverse-variance weight for exposure.

$$\beta_Y = \sum_{i=1}^n (w_{Y_i} \beta_{Y_i}) / \sum_{i=1}^n (w_{Y_i}) \quad (\text{aggregated SNP-outcome effect})$$

where $w_{Y_i} = 1 / (\beta_{Yse_i})^2$, is the inverse-variance weight for outcome.

$$\beta_{Xse} = \sqrt{[1 / \sum_{i=1}^n (w_{X_i})]} \quad (\text{SE of aggregated } \beta_X)$$

$$\beta_{Yse} = \sqrt{[1 / \sum_{i=1}^n (w_{Y_i})]} \quad (\text{SE of aggregated } \beta_Y)$$

For aggregation it assumes fixed-effects IVW (valid if heterogeneity is low; check with Cochran's Q test if needed).

(3) Computed Values

Below I extract the per-SNP values (betaX, betaY, betaXse, betaYse) directly from the data. For completeness, I also compute the aggregated values using the IVW formulas above (rounded to 3 decimals for readability; exact values depend on software precision).

The following is the Per-SNP Values Table:

SNP	betaX	betaXse	betaY	betaYse
rs9939609	0.045	0.008	0.156	0.045
rs1801133	0.032	0.007	0.089	0.038
rs7903146	0.051	0.009	0.234	0.052
rs1421085	0.038	0.006	0.178	0.041
rs1558902	0.047	0.008	0.201	0.046
rs7185735	0.029	0.007	0.112	0.039
rs71358633	0.041	0.009	0.167	0.050
rs10821905	0.033	0.006	0.134	0.037
rs116888810	0.039	0.008	0.189	0.044
rs12970134	0.036	0.007	0.145	0.040
rs17782313	0.040	0.007	0.160	0.042

The following are Aggregated Values (IVW Across All SNPs):

betaX = 0.039

betaY = 0.162

betaXse = 0.002

betaYse = 0.010

These aggregated values represent the overall weighted average effect of the SNPs on the exposure and outcome, respectively.

For the basic MR analysis, the correlation matrix in the input data is the identity matrix. For advanced MR analysis, the correlation matrix can be obtained from public databases or by AI (DeepSeek, Gemini, etc.) data construction.

3 Complete Codes

The following are the HTML+JavaScript codes of the data generator for Mendelian Randomization (MR) analysis ([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); Fig. 1-2):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>MR Data Generator</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: 1200px;
    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: 20px;
    font-size: 14px;
  }

  .workflow-section {
    margin-top: 30px;
    padding: 20px;
    border: 2px solid #e0e0e0;
    border-radius: 12px;
    background: #f9f9f9;
  }
```

```
}

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

label {
  display: block;
  margin-bottom: 8px;
  color: #333;
  font-weight: 600;
  font-size: 14px;
}

input[type="text"],
input[type="password"],
select,
textarea {
  width: 100%;
  padding: 12px;
  border: 2px solid #e0e0e0;
  border-radius: 8px;
  font-size: 14px;
  transition: border-color 0.3s;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

textarea {
  font-family: 'Courier New', monospace;
  font-size: 13px;
  resize: vertical;
  min-height: 300px;
}

textarea.output-json {
  background: #f0f0f0;
  min-height: 400px;
}

input:focus, select:focus, textarea:focus {
  outline: none;
```

```
        border-color: #667eea;
    }

    .btn {
        padding: 12px 24px;
        border: none;
        border-radius: 8px;
        font-size: 14px;
        font-weight: 600;
        cursor: pointer;
        transition: all 0.3s;
        margin-right: 10px;
        margin-bottom: 10px;
    }

    .btn-primary {
        background: #667eea;
        color: white;
    }

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

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

    .btn-success:hover:not(:disabled) {
        background: #218838;
    }

    .btn:disabled {
        background: #ccc;
        cursor: not-allowed;
        transform: none;
    }

    .file-input-group {
```

```
border: 2px dashed #ccc;
border-radius: 8px;
padding: 20px;
text-align: center;
background: #f9f9f9;
margin-top: 10px;
}

.file-input-group:hover {
border-color: #667eea;
background: #f0f7ff;
}

.status {
padding: 10px;
border-radius: 6px;
margin-top: 10px;
font-size: 14px;
}

.status.success {
background: #d4edda;
color: #155724;
border: 1px solid #c3e6cb;
}

.status.error {
background: #f8d7da;
color: #721c24;
border: 1px solid #f5c6cb;
}

.status.warning {
background: #fff3cd;
color: #856404;
border: 1px solid #ffeea7;
}

.status.info {
background: #d1ecf1;
color: #0c5460;
```

```
        border: 1px solid #bee5eb;
    }

    .two-column {
        display: grid;
        grid-template-columns: 1fr 1fr;
        gap: 20px;
    }

    .example-data {
        font-size: 11px;
        color: #666;
        margin-top: 8px;
        padding: 8px;
        background: #f8f9fa;
        border-radius: 4px;
        border-left: 3px solid #667eea;
    }

    .api-info {
        font-size: 12px;
        color: #666;
        margin-top: 5px;
        padding: 8px;
        background: #f0f7ff;
        border-radius: 4px;
    }

    .harmonization-progress {
        background: #f8f9fa;
        border: 1px solid #dee2e6;
        border-radius: 8px;
        padding: 15px;
        margin: 20px 0;
    }

    .progress-step {
        display: flex;
        align-items: center;
        margin-bottom: 10px;
        padding: 10px;
```

```
        border-radius: 6px;
    }

    .progress-step.pending {
        background: #f8f9fa;
        color: #6c757d;
    }

    .progress-step.active {
        background: #fff3cd;
        color: #856404;
    }

    .progress-step.complete {
        background: #d4edda;
        color: #155724;
    }

    .progress-step.error {
        background: #f8d7da;
        color: #721c24;
    }

    .step-icon {
        margin-right: 10px;
        font-size: 1.2rem;
    }

    .correlation-info {
        background: #e7f3ff;
        border: 1px solid #b3d9ff;
        border-radius: 8px;
        padding: 12px;
        margin: 15px 0;
        font-size: 13px;
    }

    .correlation-info strong {
        color: #004085;
    }
}
```

```
.correlation-tabs {  
  display: flex;  
  gap: 10px;  
  margin: 15px 0;  
  border-bottom: 2px solid #e0e0e0;  
}
```

```
.correlation-tab {  
  padding: 10px 15px;  
  background: #f0f0f0;  
  border: none;  
  cursor: pointer;  
  border-radius: 6px 6px 0 0;  
  font-weight: 600;  
  transition: all 0.3s;  
}
```

```
.correlation-tab.active {  
  background: #667eea;  
  color: white;  
}
```

```
.correlation-tab:hover {  
  background: #e0e0e0;  
}
```

```
.correlation-tab.active:hover {  
  background: #5568d3;  
}
```

```
.correlation-content {  
  display: none;  
  padding: 15px;  
  background: #f9f9f9;  
  border-radius: 0 8px 8px 8px;  
  margin-bottom: 15px;  
}
```

```
.correlation-content.active {  
  display: block;  
}
```

```

@media (max-width: 768px) {
  .two-column {
    grid-template-columns: 1fr;
  }

  .correlation-tabs {
    flex-direction: column;
  }
}
</style>
</head>
<body>
  <div class="container">
    <h1> MR Data Generator</h1>
    <p class="subtitle"><a
href="http://www.iaees.org/publications/journals/np/articles/2025-10(3-4)/1-Zhang-Abstract.asp">Zhang WJ. 2025. A
web-based data generator for Mendelian Randomization (MR) analysis. Network Pharmacology, 10(3-4): 14-65</a></p>

    <!-- Workflow 2: File Upload & Harmonization -->
    <div id="workflow-files" class="workflow-section">

      <!-- Step 1: File Upload -->
      <div class="form-group">
        <h3 style="color: #667eea; margin-bottom: 15px;">Step 1: Upload Files</h3>

        <div class="two-column">
          <div>
            <label for="exposureFile">Exposure File</label>
            <div class="file-input-group">
              <p> Click or drag to upload</p>
              <input type="file" id="exposureFile" accept=".csv,.txt,.json,.gz" />
            </div>
            <div class="example-data">
              <strong>CSV format:</strong> SNP,ea,nea,beta,se,eaf<br>
              <strong>JSON format:</strong> {"rs9939609": {"ea":"A","nea":"T","beta":0.15,"se":0.02}}
            </div>
            <div id="exposureStatus" class="status" style="display: none;"></div>
          </div>

          <div>

```

```

<label for="outcomeFile">Outcome File</label>
<div class="file-input-group">
  <p> Click or drag to upload</p>
  <input type="file" id="outcomeFile" accept=".csv,.txt,.json,.gz" />
</div>
<div class="example-data">
  <strong>CSV format:</strong> SNP,beta,se,ea,nea,eaf<br>
  <strong>JSON format:</strong> {"rs9939609": {"beta":-0.08,"se":0.03,"ea":"A","nea":"T"}}
</div>
<div id="outcomeStatus" class="status" style="display: none;"></div>
</div>
</div>
</div>

<!-- Step 2: Harmonization Controls -->
<div class="form-group">
  <h3 style="color: #667eea; margin-bottom: 15px;">Step 2: Harmonization & Correlation Matrix</h3>

  <div class="correlation-info">
    <strong>📌 Correlation Matrix Source:</strong> Choose between public genetic databases or
    AI-generated correlation matrices. Public databases (NIH LDlink, 1000 Genomes, gnomAD) are free. AI options (DeepSeek,
    Gemini) provide alternative correlation estimation.
  </div>

  <!-- Correlation Source Tabs -->
  <div class="correlation-tabs">
    <button class="correlation-tab active" data-source="public"> Public Databases</button>
    <button class="correlation-tab" data-source="ai"> AI-Generated</button>
  </div>

  <!-- Public Databases Content -->
  <div class="correlation-content active" id="public-correlation">
    <label for="correlationSource">Correlation Database</label>
    <select id="correlationSource">
      <option value="ldlink">NIH LDlink (Public, No Token)</option>
      <option value="1000genomes">1000 Genomes Project (Public, No Token)</option>
      <option value="gnomad">gnomAD v3.1 (Optional Token)</option>
    </select>

    <div id="tokenGroup" class="form-group" style="display: none; margin-top: 15px;">
      <label for="dbToken">Database API Token (Optional)</label>

```

```

        <input type="text" id="dbToken" placeholder="Enter API token if available" />
        <div class="api-info">
            <strong>gnomAD:</strong> <a href="https://gnomad.broadinstitute.org/"
target="_blank">Get token</a>
        </div>
    </div>
</div>

<!-- AI-Generated Content -->
<div class="correlation-content" id="ai-correlation">
    <label for="aiCorrelationService">AI Service for Correlation Matrix</label>
    <select id="aiCorrelationService">
        <option value="deepseek">DeepSeek (Recommended - Free Tier)</option>
        <option value="gemini">Google Gemini</option>
    </select>

    <div class="form-group" style="margin-top: 15px;">
        <label for="aiCorrelationApiKey">API Key</label>
        <input type="password" id="aiCorrelationApiKey" placeholder="Enter your API key" />
        <div class="api-info" id="aiCorrelationApiInfo">
            <strong>DeepSeek:</strong> <a href="https://platform.deepseek.com/" target="_blank">Get
API key</a> (Free tier available)<br>
            <strong>Gemini:</strong> <a href="https://makersuite.google.com/app/apikey"
target="_blank">Get API key</a>
        </div>
    </div>
</div>

<button class="btn btn-primary" id="harmonizeBtn" disabled style="margin-top: 15px;">
    Harmonize Data & Fetch Correlation Matrix
</button>
</div>

<!-- Harmonization Progress -->
<div id="harmonizationProgress" class="harmonization-progress" style="display: none;">
    <h4 style="margin-bottom: 10px;">Harmonization Progress:</h4>
    <div class="progress-step pending" id="step-load">
        <span class="step-icon"> </span>
        <span>Loading and parsing files...</span>
    </div>
    <div class="progress-step pending" id="step-harmonize">

```

```

        <span class="step-icon"> </span>
        <span>Harmonizing exposure and outcome data...</span>
    </div>
    <div class="progress-step pending" id="step-correlation">
        <span class="step-icon"> </span>
        <span>Fetching correlation matrix...</span>
    </div>
    <div class="progress-step pending" id="step-json">
        <span class="step-icon"> </span>
        <span>Generating final JSON output...</span>
    </div>
</div>

<!-- Output -->
<div class="form-group" id="filesOutput" style="display: none;">
    <h3 style="color: #667eea; margin-bottom: 15px;"> Final JSON Output</h3>
    <textarea id="filesJsonOutput" class="output-json" readonly></textarea>

    <div style="margin-top: 15px;">
        <button class="btn btn-success" id="copyFilesJsonBtn"> Copy JSON</button>
        <button class="btn btn-success" id="downloadFilesJsonBtn">↓ Download JSON</button>
    </div>
</div>
</div>
</div>

<!-- Dependencies -->
<script src="http://www.iaees.org/publications/software/JavaScript/pako.min.js"></script>

<script>
    // Global state
    const STATE = {
        exposureData: {},
        outcomeData: {},
        harmonizedData: null,
        correlationMatrix: null,
        finalJson: null,
        correlationSource: 'public'
    };

    // ===== Workflow 2: File Upload & Harmonization =====

```

```
function initializeFilesWorkflow() {
  const correlationSource = document.getElementById('correlationSource');
  const tokenGroup = document.getElementById('tokenGroup');
  const correlationTabs = document.querySelectorAll('.correlation-tab');
  const aiCorrelationService = document.getElementById('aiCorrelationService');

  // Correlation source database change
  correlationSource.addEventListener('change', () => {
    if (correlationSource.value === 'ldlink' || correlationSource.value === '1000genomes') {
      tokenGroup.style.display = 'none';
    } else {
      tokenGroup.style.display = 'block';
    }
  });

  // Correlation source tab switching
  correlationTabs.forEach(tab => {
    tab.addEventListener('click', () => {
      const source = tab.dataset.source;
      STATE.correlationSource = source;

      // Update tab active state
      correlationTabs.forEach(t => t.classList.remove('active'));
      tab.classList.add('active');

      // Update content visibility
      document.querySelectorAll('.correlation-content').forEach(content => {
        content.classList.remove('active');
      });
      document.getElementById(`${source}-correlation`).classList.add('active');
    });
  });

  // AI correlation service change
  aiCorrelationService.addEventListener('change', updateAICorrelationInfo);
}

function updateAICorrelationInfo() {
  const service = document.getElementById('aiCorrelationService').value;
  const info = document.getElementById('aiCorrelationApiInfo');
```

```

const infos = {
  deepseek: '<strong>DeepSeek:</strong> <a href="https://platform.deepseek.com/" target="_blank">Get API
key</a> (Free tier available)',
  gemini: '<strong>Gemini:</strong> <a href="https://makersuite.google.com/app/apikey"
target="_blank">Get API key</a>'
};

info.innerHTML = infos[service];
}

document.getElementById('exposureFile').addEventListener('change', handleExposureFile);
document.getElementById('outcomeFile').addEventListener('change', handleOutcomeFile);
document.getElementById('harmonizeBtn').addEventListener('click', harmonizeFiles);
document.getElementById('copyFilesJsonBtn').addEventListener('click', () => copyToClipboard('filesJsonOutput'));
document.getElementById('downloadFilesJsonBtn').addEventListener('click', () =>
downloadJson(document.getElementById('filesJsonOutput').value, 'mr_harmonized_data.json'));

async function handleExposureFile(e) {
  const file = e.target.files[0];
  if (!file) return;

  const status = document.getElementById('exposureStatus');
  status.textContent = ` Loading ${file.name}...`;
  status.className = 'status info';
  status.style.display = 'block';

  try {
    const data = await parseFile(file);
    STATE.exposureData = data;

    status.textContent = ` Loaded ${Object.keys(data).length} exposure variants from ${file.name}`;
    status.className = 'status success';

    checkFilesReady();
  } catch (error) {
    status.textContent = ` Error: ${error.message}`;
    status.className = 'status error';
  }
}

async function handleOutcomeFile(e) {

```

```

const file = e.target.files[0];
if (!file) return;

const status = document.getElementById('outcomeStatus');
status.textContent = ` Loading ${file.name}...`;
status.className = 'status info';
status.style.display = 'block';

try {
  const data = await parseFile(file);
  STATE.outcomeData = data;

  status.textContent = ` Loaded ${Object.keys(data).length} outcome variants from ${file.name}`;
  status.className = 'status success';

  checkFilesReady();
} catch (error) {
  status.textContent = ` Error: ${error.message}`;
  status.className = 'status error';
}
}

function checkFilesReady() {
  const harmonizeBtn = document.getElementById('harmonizeBtn');
  if (Object.keys(STATE.exposureData).length > 0 && Object.keys(STATE.outcomeData).length > 0) {
    harmonizeBtn.disabled = false;
  }
}

async function parseFile(file) {
  const buf = await file.arrayBuffer();
  let text;

  // Handle gzip
  if (file.name.endsWith('.gz') || (new Uint8Array(buf, 0, 2)[0] === 0x1f && new Uint8Array(buf, 0, 2)[1] ===
0x8b)) {
    text = pako.ungzip(new Uint8Array(buf), { to: 'string' });
  } else {
    text = new TextDecoder().decode(buf);
  }
}

```

```
// Try JSON first
if (file.name.endsWith('.json') || text.trim().startsWith('{')) {
  try {
    const parsed = JSON.parse(text);
    return normalizeJsonData(parsed);
  } catch (e) {
    // Not JSON, fall through to CSV
  }
}

// Parse CSV/TSV
return parseCsvData(text);
}

function normalizeJsonData(parsed) {
  const result = {};

  if (Array.isArray(parsed)) {
    parsed.forEach(row => {
      const normalized = normalizeRow(row);
      if (normalized.rsid) {
        result[normalized.rsid] = normalized;
      }
    });
  } else if (typeof parsed === 'object') {
    Object.entries(parsed).forEach(([key, value]) => {
      const normalized = normalizeRow({ ...value, rsid: key });
      if (normalized.rsid) {
        result[normalized.rsid] = normalized;
      }
    });
  }

  return result;
}

function parseCsvData(text) {
  const lines = text.split(/\r?\n/).filter(l => l.trim());
  if (lines.length < 2) throw new Error('File is empty or has no data rows');

  const firstLine = lines[0];
```

```

const delim = firstLine.includes('\t') ? '\t' : ',';
const headers = firstLine.split(delim).map(h => h.trim().replace(/^[^|"]$/g, "").toLowerCase());

const result = {};

for (let i = 1; i < lines.length; i++) {
  const parts = lines[i].split(delim).map(p => p.trim().replace(/^[^|"]$/g, ""));
  const obj = {};

  headers.forEach((header, idx) => {
    if (idx < parts.length) {
      obj[header] = parts[idx];
    }
  });

  const normalized = normalizeRow(obj);
  if (normalized.rsid) {
    result[normalized.rsid] = normalized;
  }
}

return result;
}

function normalizeRow(row) {
  const fieldMap = {
    rsid: ['snp', 'rsid', 'variant', 'markername', 'id', 'snpid'],
    ea: ['effect_allele', 'ea', 'a1', 'effectallele', 'allele1'],
    nea: ['other_allele', 'nea', 'a2', 'otherallele', 'allele2'],
    beta: ['beta', 'b', 'beta_estimate', 'effect_size', 'betavalue'],
    se: ['se', 'stderr', 'beta_se', 'betavalue'],
    eaf: ['eaf', 'maf', 'allele_frequency', 'af', 'frequency']
  };

  const lowerRow = {};
  Object.entries(row).forEach(([k, v]) => {
    lowerRow[k.toLowerCase()] = v;
  });

  const result = {};
  Object.entries(fieldMap).forEach(([target, keys]) => {

```

```

    for (const key of keys) {
      if (key in lowerRow && lowerRow[key] != null && lowerRow[key] !== "") {
        result[target] = lowerRow[key];
        break;
      }
    }
  });

  // Convert numbers
  if (result.beta) result.beta = Number(result.beta);
  if (result.se) result.se = Number(result.se);
  if (result.eaf) result.eaf = Number(result.eaf);

  return result;
}

async function harmonizeFiles() {
  const progress = document.getElementById('harmonizationProgress');
  progress.style.display = 'block';

  try {
    // Step 1: Already loaded
    updateProgressStep('step-load', 'complete', ' Files loaded successfully');

    // Step 2: Harmonize
    updateProgressStep('step-harmonize', 'active', ' Harmonizing data...');
    await new Promise(resolve => setTimeout(resolve, 500));

    const harmonized = harmonizeData(STATE.exposureData, STATE.outcomeData);
    STATE.harmonizedData = harmonized;

    updateProgressStep('step-harmonize', 'complete', ` Harmonized ${harmonized.snps.length} common
SNPs`);

    // Step 3: Fetch correlation
    updateProgressStep('step-correlation', 'active', ' Fetching correlation matrix...');

    let correlation;
    if (STATE.correlationSource === 'public') {
      const source = document.getElementById('correlationSource').value;
      const token = document.getElementById('dbToken').value.trim();

```

```

        correlation = await fetchCorrelationMatrix(harmonized.snps, source, token);
    } else {
        const service = document.getElementById('aiCorrelationService').value;
        const apiKey = document.getElementById('aiCorrelationApiKey').value.trim();

        if (!apiKey) {
            throw new Error('Please provide an API key for AI correlation matrix generation');
        }

        correlation = await fetchAICorrelationMatrix(harmonized.snps, service, apiKey);
    }

    STATE.correlationMatrix = correlation;

    if (correlation.source === 'identity') {
        updateProgressStep('step-correlation', 'complete', ' Database unavailable - using identity matrix');
    } else {
        updateProgressStep('step-correlation', 'complete', ` Correlation matrix fetched from
    ${correlation.source}`);
    }

    // Step 4: Generate JSON
    updateProgressStep('step-json', 'active', ' Generating final JSON...');
    await new Promise(resolve => setTimeout(resolve, 300));

    const finalJson = {
        snps: harmonized.snps,
        betaX: harmonized.betaX,
        betaY: harmonized.betaY,
        betaXse: harmonized.betaXse,
        betaYse: harmonized.betaYse,
        correlation: correlation.matrix,
        exposure: "Exposure",
        outcome: "Outcome",
        n: harmonized.snps.length,
        metadata: {
            harmonized_snps: harmonized.snps.length,
            correlation_source: correlation.source,
            correlation_method: correlation.method,
            timestamp: new Date().toISOString()
        }
    }

```

```

    };

    STATE.finalJson = finalJson;

    updateProgressStep('step-json', 'complete', 'JSON generated successfully');

    // Display output
    document.getElementById('filesJsonOutput').value = JSON.stringify(finalJson, null, 2);
    document.getElementById('filesOutput').style.display = 'block';

} catch (error) {
    console.error('Harmonization error:', error);
    const activeStep = document.querySelector('.progress-step.active');
    if (activeStep) {
        updateProgressStep(activeStep.id, 'error', `Error: ${error.message}`);
    }
}

function harmonizeData(exposureData, outcomeData) {
    const commonRsids = Object.keys(exposureData).filter(rsid => outcomeData[rsid]);

    const snps = [];
    const betaX = [];
    const betaY = [];
    const betaXse = [];
    const betaYse = [];

    for (const rsid of commonRsids) {
        const exp = exposureData[rsid];
        const out = outcomeData[rsid];

        // Simple allele alignment
        let betaYValue = out.beta;

        // Flip if alleles are reversed
        if (exp.ea && exp.nea && out.ea && out.nea) {
            if (exp.ea === out.nea && exp.nea === out.ea) {
                betaYValue = -betaYValue;
            }
        }
    }
}

```

```

    snps.push(rsid);
    betaX.push(exp.beta);
    betaY.push(betaYValue);
    betaXse.push(exp.se);
    betaYse.push(out.se);
  }

  return { snps, betaX, betaY, betaXse, betaYse };
}

async function fetchCorrelationMatrix(snps, source, token) {
  try {
    // Try to fetch from actual database
    if (source === 'ldlink') {
      return await fetchLDlinkCorrelation(snps);
    } else if (source === '1000genomes') {
      return await fetch1000GenomesCorrelation(snps);
    } else if (source === 'gnomad') {
      return await fetchGnomADCorrelation(snps, token);
    }
  } catch (error) {
    console.warn(`Failed to fetch from ${source}:`, error);
  }

  // Fallback to identity matrix
  return createIdentityMatrix(snps);
}

async function fetchLDlinkCorrelation(snps) {
  // LDlink API implementation
  const snpSubset = snps.slice(0, 10); // Limit for API

  try {
    const snpString = snpSubset.join('%2B');
    const url = `https://ldlink.nih.gov/LDlinkRest/ldmatrix?snps=${snpString}&pop=CEU&r2_d=r2`;

    const response = await fetch(url, {
      headers: { 'Accept': 'text/plain' }
    });
  }
}

```

```
    if (response.ok) {
      const text = await response.text();
      const matrix = parseLDMatrix(text, snpSubset.length);

      // Extend matrix to full size with identity for remaining SNPs
      const fullMatrix = extendMatrixToFullSize(matrix, snps.length);

      return {
        matrix: fullMatrix,
        source: 'ldlink',
        method: 'ldlink_api'
      };
    }
  } catch (error) {
    console.warn('LDlink fetch failed:', error);
  }

  throw new Error('LDlink fetch failed');
}

async function fetch1000GenomesCorrelation(snps) {
  // Simplified: Try LDlink API as fallback
  const snpSubset = snps.slice(0, 10); // Limit for API

  try {
    const snpString = snpSubset.join('%2B');
    const url = `https://ldlink.nih.gov/LDlinkRest/ldmatrix?snp=${snpString}&pop=CEU&r2_d=r2`;

    const response = await fetch(url, {
      headers: { 'Accept': 'text/plain' }
    });

    if (response.ok) {
      const text = await response.text();
      const matrix = parseLDMatrix(text, snpSubset.length);

      // Extend matrix to full size with identity for remaining SNPs
      const fullMatrix = extendMatrixToFullSize(matrix, snps.length);

      return {
        matrix: fullMatrix,
```

```

        source: '1000genomes',
        method: 'ldlink_api'
    };
}
} catch (error) {
    console.warn('1000 Genomes fetch failed:', error);
}

throw new Error('1000 Genomes fetch failed');
}

async function fetchGnomADCorrelation(snps, token) {
    // Placeholder - would need actual gnomAD API implementation
    throw new Error('gnomAD API not available');
}

async function fetchAICorrelationMatrix(snps, service, apiKey) {
    const snpList = snps.join(', ');
    const prompt = `You are a genetics expert. For the following SNPs, generate a realistic genetic correlation matrix
based on typical linkage disequilibrium patterns.

```

SNPs: \${snpList}

Generate a JSON response with ONLY the following format (no markdown, no extra text):

```

{
  "matrix": [[1.0, 0.15, 0.08, ...], [0.15, 1.0, 0.12, ...], ...],
  "description": "LD-based correlation matrix"
}

```

Rules:

1. Diagonal elements must be 1.0
2. Matrix must be symmetric
3. All values between -1 and 1
4. Use realistic LD patterns (R^2 values converted to correlation)
5. Nearby SNPs should have higher correlation`;

```

    try {
        if (service === 'deepseek') {
            const 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. Respond with valid JSON
only.' },
            { role: 'user', content: prompt }
        ],
        temperature: 0.5,
        max_tokens: 2000
    })
});

if (!response.ok) {
    const error = await response.json();
    throw new Error(error.error?.message || `API error: ${response.status}`);
}

const data = await response.json();
const content = data.choices?.[0]?.message?.content;

if (!content) {
    throw new Error('No content received from AI');
}

return parseAICorrelationResponse(content, snps);

} else if (service === 'gemini') {
    const response = await
fetch(`https://generativelanguage.googleapis.com/v1/models/gemini-1.5-flash:generateContent?key=${apiKey}`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
            contents: [{ parts: [{ text: prompt }] }],
            generationConfig: { temperature: 0.5, maxOutputTokens: 2000 }
        })
    });
});

if (!response.ok) {

```

```

        const error = await response.json();
        throw new Error(error.error?.message || `API error: ${response.status}`);
    }

    const data = await response.json();
    const content = data.candidates?.[0]?.content?.parts?.[0]?.text;

    if (!content) {
        throw new Error('No content received from AI');
    }

    return parseAICorrelationResponse(content, snps);
}

} catch (error) {
    console.warn(`AI correlation fetch failed: ${error.message}`);
}

throw new Error(`AI correlation matrix generation failed`);
}

function parseAICorrelationResponse(content, snps) {
    try {
        // Extract JSON from response
        const jsonMatch = content.match(/\{[\s\S]*\}/);
        if (!jsonMatch) {
            throw new Error('No JSON found in response');
        }

        const parsed = JSON.parse(jsonMatch[0]);

        if (!parsed.matrix || !Array.isArray(parsed.matrix)) {
            throw new Error('Invalid matrix format in response');
        }

        // Validate matrix dimensions
        if (parsed.matrix.length !== snps.length) {
            throw new Error(`Matrix dimension mismatch: expected ${snps.length}, got ${parsed.matrix.length}`);
        }

        for (let i = 0; i < parsed.matrix.length; i++) {

```

```

        if (!Array.isArray(parsed.matrix[i]) || parsed.matrix[i].length !== snps.length) {
            throw new Error(`Row ${i} has incorrect length`);
        }
    }

    return {
        matrix: parsed.matrix,
        source: 'ai_deepseek' || 'ai_gemini',
        method: 'ld_based_ai_estimation'
    };

} catch (error) {
    throw new Error(`Failed to parse AI correlation response: ${error.message}`);
}

}

function parseLDMatrix(text, size) {
    const lines = text.trim().split("\n");
    const matrix = [];

    for (let i = 1; i <= size && i < lines.length; i++) {
        const parts = lines[i].split("\t");
        const row = parts.slice(1, size + 1).map(v => {
            const num = parseFloat(v);
            return isNaN(num) ? 0 : Math.sqrt(Math.max(0, num)); // R2 to r
        });
        matrix.push(row);
    }

    return matrix;
}

function extendMatrixToFullSize(partialMatrix, fullSize) {
    const matrix = [];

    for (let i = 0; i < fullSize; i++) {
        const row = [];
        for (let j = 0; j < fullSize; j++) {
            if (i < partialMatrix.length && j < partialMatrix[i].length) {
                row.push(partialMatrix[i][j]);
            } else {

```

```
        row.push(i === j ? 1.0 : 0.0);
    }
}
matrix.push(row);
}

return matrix;
}

function createIdentityMatrix(snps) {
    const n = snps.length;
    const matrix = [];

    for (let i = 0; i < n; i++) {
        const row = [];
        for (let j = 0; j < n; j++) {
            row.push(i === j ? 1.0 : 0.0);
        }
        matrix.push(row);
    }

    return {
        matrix: matrix,
        source: 'identity',
        method: 'fallback'
    };
}

function updateProgressStep(stepId, status, text) {
    const step = document.getElementById(stepId);
    if (!step) return;

    step.className = `progress-step ${status}`;

    const icons = {
        pending: ' ',
        active: ' ',
        complete: ' ',
        error: ' '
    };
};
```

```
step.querySelector('.step-icon').textContent = icons[status] || ' ';
step.querySelector('span:last-child').textContent = text;
}

// ===== Utility Functions =====
function copyToClipboard(textareaId) {
    const textarea = document.getElementById(textareaId);
    textarea.select();
    document.execCommand('copy');

    // Show feedback
    const btn = event.target;
    const originalText = btn.textContent;
    btn.textContent = '✓ Copied!';
    setTimeout(() => {
        btn.textContent = originalText;
    }, 2000);
}

function downloadJson(jsonString, filename) {
    const blob = new Blob([jsonString], { type: 'application/json' });
    const url = URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = filename;
    document.body.appendChild(a);
    a.click();
    document.body.removeChild(a);
    URL.revokeObjectURL(url);
}

// Initialize on page load
document.addEventListener('DOMContentLoaded', initializeFilesWorkflow);
</script>
</body>
</html>
```

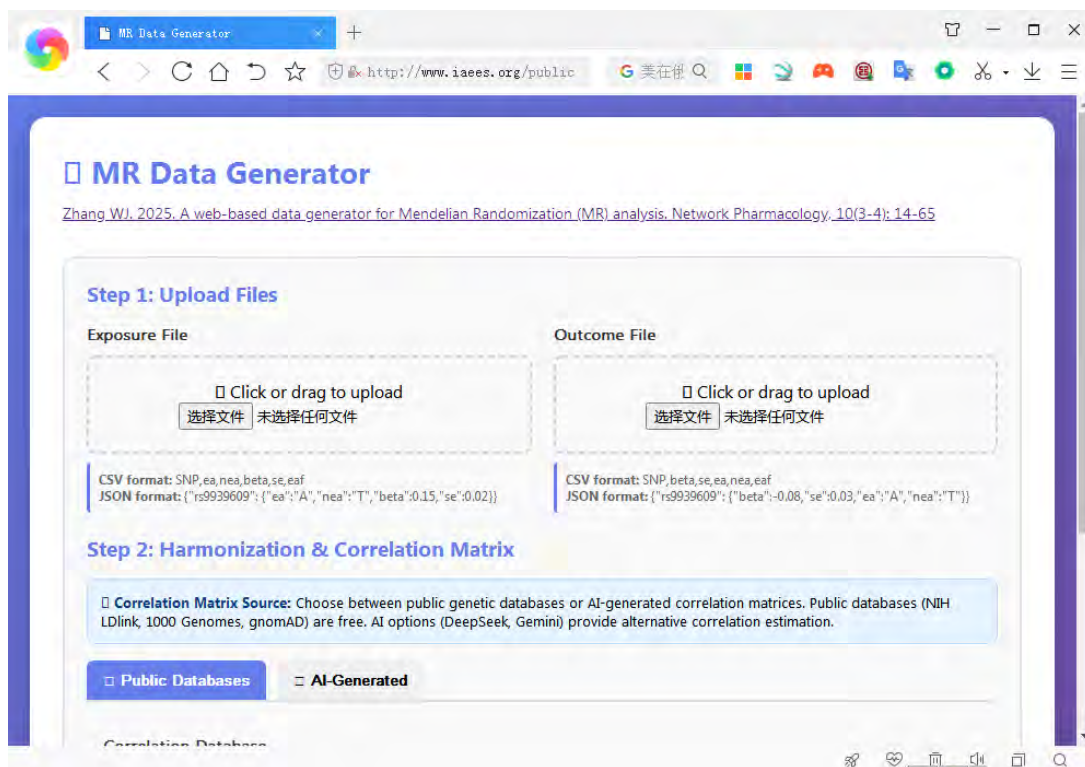


Fig. 1 Web-based data generator (homepage).

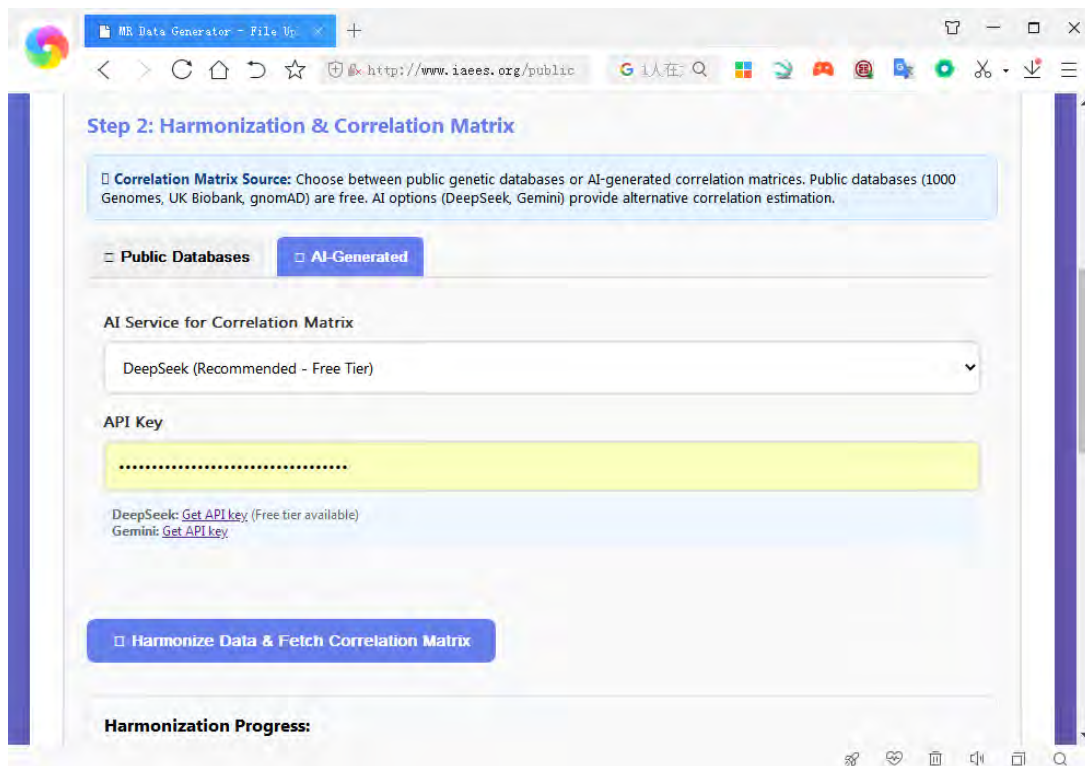


Fig. 2 Web-based data generator (AI correlation matrix construction).

4 Algorithmic Description

4.1 Overview

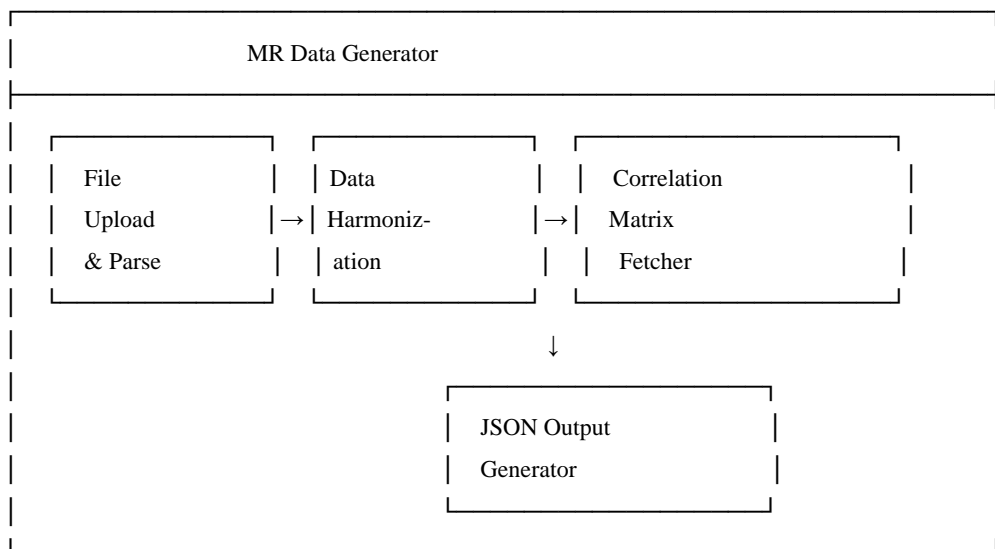
MR Data Generator is a web-based tool for harmonizing genetic variant data from exposure and outcome datasets for Mendelian Randomization (MR) analysis. The application processes genetic association data files, harmonizes alleles, and generates correlation matrices to account for linkage disequilibrium (LD) between variants.

4.2 Key Features:

- Upload and parse exposure/outcome files (CSV, TSV, JSON, GZIP)
- Automatic allele harmonization
- Multiple correlation matrix sources (public databases and AI-generated)
- Export harmonized data as JSON

4.3 Algorithmic Description

(1) System Architecture



(2) File Parsing Algorithm

(2.1) File Format Detection

Input: File buffer (ArrayBuffer)

Output: Parsed genetic variant data (Object)

ALGORITHM: parseFile(file)

INPUT: file - File object from user upload

OUTPUT: Object mapping rsIDs to variant data

1. READ file into ArrayBuffer (buf)
2. IF file is GZIP compressed THEN
 - a. DETECT: Check first 2 bytes == [0x1f, 0x8b]
 - b. DECOMPRESS using pako.ungzip()
 - c. SET text = decompressed string
3. ELSE
 - a. SET text = decode buf as UTF-8
4. END IF

```
5. TRY to parse as JSON:
  IF text starts with '{' OR filename ends with '.json' THEN
    a. parsed = JSON.parse(text)
    b. RETURN normalizeJsonData(parsed)
  END IF
```

```
6. CATCH: If JSON parsing fails, continue
```

```
7. PARSE as CSV/TSV:
  a. RETURN parseCsvData(text)
```

(2.2) CSV/TSV Parsing

ALGORITHM: parseCsvData(text)

INPUT: text - Raw CSV/TSV content

OUTPUT: Object mapping rsIDs to variant data

```
1. SPLIT text by newlines → lines[]
2. IF lines.length < 2 THEN
  a. THROW "File is empty or has no data rows"
3. END IF
```

```
4. DETECT delimiter:
  IF firstLine contains '\t' THEN
    delim = '\t'
  ELSE
    delim = ','
  END IF
```

```
5. PARSE headers:
  headers = firstLine.split(delim).map(toLowercase)
```

```
6. INITIALIZE result = {}
```

```
7. FOR EACH line in lines[1:] DO
  a. parts = line.split(delim)
  b. obj = {} // Map headers to values
  c. FOR EACH header, index in headers DO
    i. obj[header] = parts[index]
  d. END FOR

  e. normalized = normalizeRow(obj)
  f. IF normalized.rsid EXISTS THEN
    i. result[normalized.rsid] = normalized
  g. END IF
8. END FOR
```

9. RETURN result

(2.3) Field Normalization

ALGORITHM: normalizeRow(row)

INPUT: row - Object with variant data (arbitrary field names)

OUTPUT: Standardized variant object

DEFINE fieldMap:

```

rsid  → ['snp', 'rsid', 'variant', 'markername', 'id']
ea    → ['effect_allele', 'ea', 'a1', 'effectallele']
nea   → ['other_allele', 'nea', 'a2', 'otherallele']
beta  → ['beta', 'b', 'beta_estimate', 'effect_size']
se    → ['se', 'stderr', 'beta_se']
eaf   → ['eaf', 'maf', 'allele_frequency', 'af']

```

1. CONVERT all keys in row to lowercase → lowerRow

2. INITIALIZE result = { }

3. FOR EACH target field in fieldMap DO

a. FOR EACH possible key in fieldMap[target] DO

i. IF key exists in lowerRow AND value is not empty THEN

- result[target] = lowerRow[key]

- BREAK

ii. END IF

b. END FOR

4. END FOR

5. CONVERT numeric fields:

- result.beta = parseFloat(result.beta)

- result.se = parseFloat(result.se)

- result.eaf = parseFloat(result.eaf)

6. RETURN result

(3) Data Harmonization Algorithm

ALGORITHM: harmonizeData(exposureData, outcomeData)

INPUT:

- exposureData: Object {rsID: {beta, se, ea, nea, ...}}

- outcomeData: Object {rsID: {beta, se, ea, nea, ...}}

OUTPUT: Harmonized data object

1. FIND common SNPs:

commonRsids = intersection(keys(exposureData), keys(outcomeData))

2. INITIALIZE output arrays:

```
snps = []
betaX = []
betaY = []
betaXse = []
betaYse = []
```

3. FOR EACH rsid in commonRsids DO

```
a. exp = exposureData[rsid]
b. out = outcomeData[rsid]

c. SET betaYValue = out.beta

d. PERFORM allele alignment:
  IF exp.ea AND exp.nea AND out.ea AND out.nea THEN
    IF exp.ea == out.nea AND exp.nea == out.ea THEN
      // Alleles are flipped
      betaYValue = -betaYValue
    END IF
  END IF

e. APPEND to output arrays:
  snps.push(rsid)
  betaX.push(exp.beta)
  betaY.push(betaYValue)
  betaXse.push(exp.se)
  betaYse.push(out.se)
```

4. END FOR

5. RETURN {snps, betaX, betaY, betaXse, betaYse}

Allele Harmonization Logic:

- **Case 1:** Effect alleles match → No change
- **Case 2:** Effect allele in exposure = Non-effect allele in outcome → Flip beta sign
- **Case 3:** Missing allele information → No flipping

(4) Correlation Matrix Fetching

(4.1) Main Fetching Logic

ALGORITHM: fetchCorrelationMatrix(snps, source, token)

INPUT:

- snps: Array of rsIDs
- source: 'ldlink' | '1000genomes' | 'gnomad'
- token: Optional API token

OUTPUT: Correlation matrix object

1. TRY:

```
a. IF source == 'ldlink' THEN
```

```

    RETURN fetchLDlinkCorrelation(snps)
  b. ELSE IF source == '1000genomes' THEN
    RETURN fetch1000GenomesCorrelation(snps)
  c. ELSE IF source == 'gnomad' THEN
    RETURN fetchGnomADCorrelation(snps, token)
  d. END IF
2. CATCH error:
  a. LOG warning
3. END TRY

```

4. FALLBACK:

```
RETURN createIdentityMatrix(snps)
```

(4.2) LDlink API Integration

ALGORITHM: fetchLDlinkCorrelation(snps)

INPUT: snps - Array of rsIDs

OUTPUT: Correlation matrix object

1. LIMIT snps to first 10 (API constraint)

```
snpSubset = snps[0:10]
```

2. BUILD API URL:

```
snpString = join(snpSubset, '%2B')
```

```
url = "https://ldlink.nih.gov/LDlinkRest/ldmatrix?"
```

```
snps={snpString}&pop=CEU&r2_d=r2"
```

3. SEND HTTP GET request to url

4. IF response is OK THEN

```
a. text = response.text
```

```
b. matrix = parseLDMatrix(text, snpSubset.length)
```

```
c. fullMatrix = extendMatrixToFullSize(matrix, snps.length)
```

```
d. RETURN {
```

```
  matrix: fullMatrix,
```

```
  source: 'ldlink',
```

```
  method: 'ldlink_api'
```

```
}
```

5. ELSE

```
a. THROW error
```

6. END IF

(4.3) LD Matrix Parsing

ALGORITHM: parseLDMatrix(text, size)

INPUT:

- text: Tab-delimited LD matrix from LDlink

- size: Number of SNPs

OUTPUT: 2D array (correlation matrix)

1. SPLIT text by newlines → lines[]
2. INITIALIZE matrix = []
3. FOR i = 1 TO size DO
 - a. parts = lines[i].split('\t')
 - b. row = []
 - c. FOR j = 1 TO size DO
 - i. value = parseFloat(parts[j])
 - ii. IF value is NaN THEN value = 0
 - iii. correlation = sqrt(max(0, value)) // Convert R² to r
 - iv. row.push(correlation)
 - d. END FOR
 - e. matrix.push(row)
4. END FOR

5. RETURN matrix

Note: LDlink returns R² values, which are converted to correlation coefficients using: $r = \sqrt{R^2}$

(4.4) Matrix Extension

ALGORITHM: extendMatrixToFullSize(partialMatrix, fullSize)

INPUT:

- partialMatrix: Small correlation matrix
- fullSize: Target matrix size

OUTPUT: Extended matrix with identity padding

1. INITIALIZE matrix = []
2. FOR i = 0 TO fullSize-1 DO
 - a. row = []
 - b. FOR j = 0 TO fullSize-1 DO
 - IF i < partialMatrix.length AND j < partialMatrix[i].length THEN
 - row.push(partialMatrix[i][j])
 - ELSE
 - row.push(i == j ? 1.0 : 0.0) // Identity for new SNPs
 - END IF
 - c. END FOR
 - d. matrix.push(row)
3. END FOR

4. RETURN matrix

(4.5) AI-Generated Correlation Matrix

ALGORITHM: fetchAICorrelationMatrix(snps, service, apiKey)

INPUT:

- snps: Array of rsIDs
- service: 'deepseek' | 'gemini'
- apiKey: API authentication key

OUTPUT: Correlation matrix object

1. BUILD prompt:

"You are a genetics expert. For the following SNPs,
generate a realistic genetic correlation matrix...
SNPs: {snps}

Rules:

1. Diagonal = 1.0
2. Symmetric matrix
3. Values in [-1, 1]
4. Use realistic LD patterns"

2. IF service == 'deepseek' THEN

- a. SEND POST to 'https://api.deepseek.com/chat/completions'
- b. HEADERS: Authorization: Bearer {apiKey}
- c. BODY: {
 - model: 'deepseek-chat',
 - messages: [prompt],
 - temperature: 0.5

3. ELSE IF service == 'gemini' THEN

- a. SEND POST to Google Gemini API
- b. Similar structure

4. END IF

5. PARSE response:

content = extractJSONFromResponse(response)

6. VALIDATE matrix:

- a. CHECK dimensions match snps.length
- b. CHECK diagonal == 1.0
- c. CHECK symmetry

7. RETURN {

- matrix: parsed.matrix,
- source: 'ai_{service}',
- method: 'ld_based_ai_estimation'

(4.6) Identity Matrix Fallback

ALGORITHM: createIdentityMatrix(snps)

INPUT: snps - Array of rsIDs

OUTPUT: Identity matrix

```

1. n = snps.length
2. INITIALIZE matrix = []

3. FOR i = 0 TO n-1 DO
  a. row = []
  b. FOR j = 0 TO n-1 DO
    row.push(i == j ? 1.0 : 0.0)
  c. END FOR
  d. matrix.push(row)
4. END FOR

5. RETURN {
  matrix: matrix,
  source: 'identity',
  method: 'fallback'
}

```

(5) JSON Output Generation

ALGORITHM: generateFinalJSON(harmonizedData, correlationMatrix)

INPUT:

- harmonizedData: {snps, betaX, betaY, betaXse, betaYse}
- correlationMatrix: {matrix, source, method}

OUTPUT: JSON object for MR analysis

```

1. CONSTRUCT JSON object:
{
  "snps": harmonizedData.snps,
  "betaX": harmonizedData.betaX,
  "betaY": harmonizedData.betaY,
  "betaXse": harmonizedData.betaXse,
  "betaYse": harmonizedData.betaYse,
  "correlation": correlationMatrix.matrix,
  "exposure": "Exposure",
  "outcome": "Outcome",
  "n": harmonizedData.snps.length,
  "metadata": {
    "harmonized_snps": harmonizedData.snps.length,
    "correlation_source": correlationMatrix.source,
    "correlation_method": correlationMatrix.method,
    "timestamp": ISO8601_timestamp
  }
}

```

2. RETURN JSON object

5 User Guide

Step-by-Step Instructions

Step 1: Prepare Your Data Files

Supported Formats:

- **CSV** (comma-separated values)
- **TSV** (tab-separated values)
- **JSON** (structured format)
- **GZIP** (compressed files: .gz)

Required Fields:

Field	Alternative Names	Description	Example
SNP/rsID	snp, rsid, variant, markername, id	SNP identifier	rs9939609
Effect Allele (EA)	effect_allele, ea, a1	Effect allele	A
Other Allele (NEA)	other_allele, nea, a2	Non-effect allele	T
Beta	beta, b, effect_size	Effect size	0.15
SE	se, stderr, beta_se	Standard error	0.02
EAF	eaf, maf, allele_frequency	Effect allele frequency	0.42

Example CSV Format:

```
csv
SNP,ea,nea,beta,se,eaf
rs9939609,A,T,0.15,0.02,0.42
rs1421085,C,T,0.12,0.018,0.38
rs17782313,C,T,0.08,0.015,0.24
```

Example JSON Format:

```
json
{
  "rs9939609": {
    "ea": "A",
    "nea": "T",
    "beta": 0.15,
    "se": 0.02,
    "eaf": 0.42
  },
  "rs1421085": {
    "ea": "C",
    "nea": "T",
    "beta": 0.12,
    "se": 0.018,
    "eaf": 0.38
  }
}
```

Step 2: Upload Files

1. **Navigate to the application** in your web browser
2. **Locate the "Step 1: Upload Files" section**
3. **Upload Exposure File:**
 - Click the " Click or drag to upload" box
 - Select your exposure data file
 - Wait for confirmation: " Loaded X exposure variants"
4. **Upload Outcome File:**
 - Repeat for outcome data
 - Wait for confirmation: " Loaded X outcome variants"

File Validation:

- The system automatically detects file format
- Parses compressed GZIP files
- Normalizes field names
- Validates data types

Step 3: Choose Correlation Matrix Source**Option A: Public Databases (Free)**

1. **Click the " Public Databases" tab** (selected by default)
2. **Select a database** from the dropdown:

Database	Description	Token Required	SNP Limit
NIH LDlink	Recommended, most reliable	No	10 SNPs
1000 Genomes	Uses LDlink API	No	10 SNPs
gnomAD v3.1	High-quality, optional token	Optional	Varies

3. **For gnomAD:**
 - Optionally enter your API token
 - Get token at: <https://gnomad.broadinstitute.org/>

Option B: AI-Generated (Requires API Key)

1. **Click the " AI-Generated" tab**
2. **Select AI service:**

Service	Description	Cost	Quality
DeepSeek	Recommended, free tier available	Free tier: 10M tokens/month	Good
Google Gemini	Alternative option	Pay-per-use	Good

3. **Enter your API key:**
 - **DeepSeek:** Get at <https://platform.deepseek.com/>
 - **Gemini:** Get at <https://makersuite.google.com/app/apikey>

Step 4: Harmonize Data

1. **Click the " Harmonize Data & Fetch Correlation Matrix" button**
2. **Monitor progress** through 4 stages:
 - Loading and parsing files...
 - Harmonizing exposure and outcome data...
 - Fetching correlation matrix...
 - Generating final JSON output...

3. **Wait for completion:**

- All stages show when complete
- Any errors show with explanation

Harmonization Process:

- Finds common SNPs between exposure and outcome
- Aligns alleles (flips beta signs when necessary)
- Fetches correlation matrix from selected source
- Generates final JSON output

Step 5: Review and Export Results

1. **Review the JSON output** in the text area

2. **Verify metadata:**

- Number of harmonized SNPs
- Correlation source used
- Timestamp

Example Output Structure:

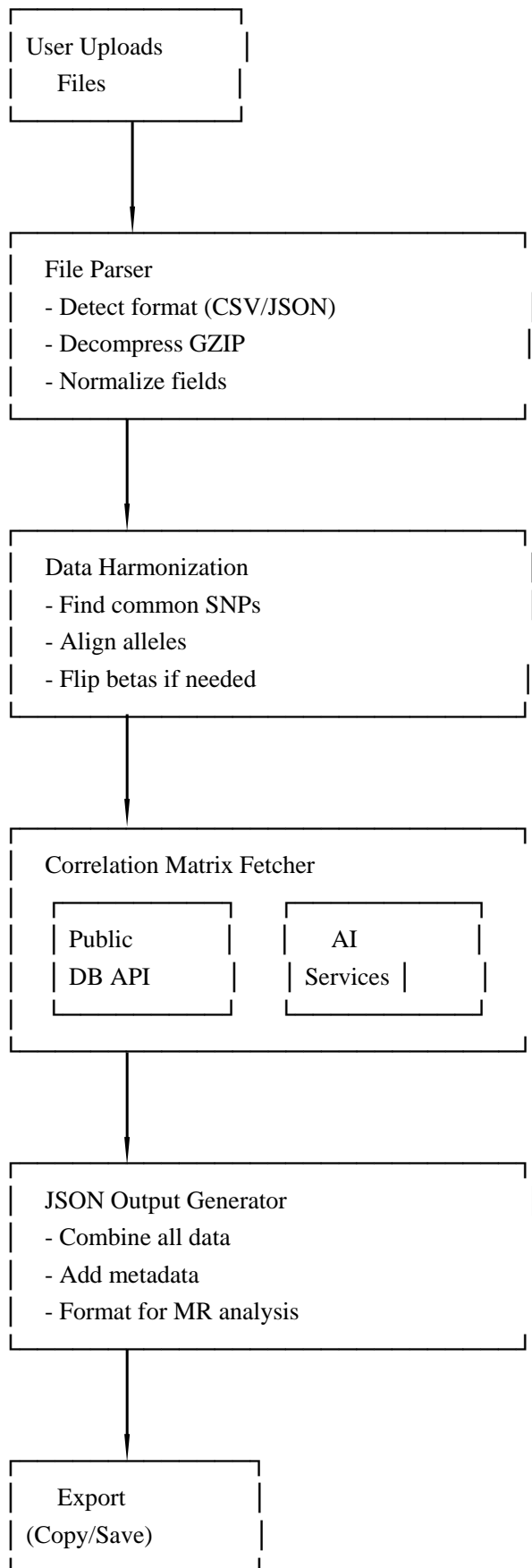
```
json
{
  "snps": ["rs9939609", "rs1421085"],
  "betaX": [0.15, 0.12],
  "betaY": [-0.08, -0.06],
  "betaXse": [0.02, 0.018],
  "betaYse": [0.03, 0.025],
  "correlation": [
    [1.0, 0.15],
    [0.15, 1.0]
  ],
  "exposure": "Exposure",
  "outcome": "Outcome",
  "n": 2,
  "metadata": {
    "harmonized_snps": 2,
    "correlation_source": "ldlink",
    "correlation_method": "ldlink_api",
    "timestamp": "2025-01-15T10:30:00.000Z"
  }
}
```

3. **Export options:**

- **Copy JSON:** Copies to clipboard
- **Download JSON:** Saves as mr_harmonized_data.json

6 Technical Specifications

Data Flow Diagram



API Endpoints Used

(1) NIH LDlink API

- **Endpoint:** <https://ldlink.nih.gov/LDlinkRest/ldmatrix>
- **Method:** GET
- **Parameters:**
 - snps: SNP list (max 10, joined with %2B)
 - pop: Population code (default: CEU)
 - r2_d: Return R² or D' (default: r2)
- **Response:** Tab-delimited LD matrix
- **Rate Limit:** Public access, no strict limits

(2) DeepSeek API

- **Endpoint:** <https://api.deepseek.com/chat/completions>
- **Method:** POST
- **Headers:**
 - Content-Type: application/json
 - Authorization: Bearer {API_KEY}
- **Body:**

```
json
{
  "model": "deepseek-chat",
  "messages": [...],
  "temperature": 0.5,
  "max_tokens": 2000
}
```

- **Rate Limit:** Free tier - 10M tokens/month

(3) Google Gemini API

- **Endpoint:** <https://generativelanguage.googleapis.com/v1/models/gemini-1.5-flash:generateContent>
- **Method:** POST
- **Parameters:** key={API_KEY} (query string)
- **Body:**

```
json
{
  "contents": [...],
  "generationConfig": {
    "temperature": 0.5,
    "maxOutputTokens": 2000
  }
}
```

Browser Compatibility

Browser Minimum Version Status

Chrome	90+	Fully Supported
Firefox	88+	Fully Supported
Safari	14+	Fully Supported

Browser Minimum Version Status

Edge	90+	Fully Supported
Opera	76+	Fully Supported

Required Features:

- ES6 JavaScript
- Fetch API
- FileReader API
- ArrayBuffer support

Performance Characteristics

Operation	Typical Time Notes	
File parsing (1000 SNPs)	< 100ms	Depends on file size
Harmonization (1000 SNPs)	< 50ms	CPU-bound
LDlink API call	2-5s	Network-dependent
AI correlation generation	5-15s	Depends on SNP count
JSON generation	< 10ms	Fast

Memory Usage:

- Typical: 10-50 MB
- Large datasets (>10,000 SNPs): Up to 200 MB

7 Troubleshooting**Common Issues and Solutions****Issue 1: File Upload Fails****Symptoms:**

- Error message: " Error: File is empty or has no data rows"
- No data loaded after upload

Solutions:

- 1. Check file format:**
 - Ensure headers are present in CSV/TSV
 - Verify JSON is valid (use JSONLint.com)
- 2. Check file encoding:**
 - Must be UTF-8
 - Avoid Excel-specific encodings
- 3. Check for required fields:**
 - At minimum: SNP/rsID, beta, se
 - Effect/non-effect alleles recommended

Issue 2: Harmonization Returns Few SNPs**Symptoms:**

- " Harmonized 5 common SNPs" when expecting more

Causes & Solutions:

- 1. Different SNP naming:**
 - Exposure uses rs123, outcome uses chr1:12345
 - **Solution:** Standardize to rsIDs
- 2. No overlap:**

- Different SNP sets entirely
- **Solution:** Use instruments present in both datasets
- 3. **Case sensitivity:**
 - Should be handled automatically
 - **Solution:** Check for typos in SNP IDs

Issue 3: Correlation Matrix Fetch Fails

Symptoms:

- " Database unavailable - using identity matrix"
- Error in correlation fetching step

Solutions:

For Public Databases:

1. **LDlink timeout:**
 - Too many SNPs (>10)
 - **Solution:** Application automatically limits to 10
2. **Network issues:**
 - Check internet connection
 - Try again after a few seconds
3. **API temporarily down:**
 - Falls back to identity matrix
 - **Solution:** Use AI-generated option

For AI Services:

1. **Invalid API key:**
 - Error: "API error: 401"
 - **Solution:** Verify API key is correct
2. **Rate limit exceeded:**
 - Error: "API error: 429"
 - **Solution:** Wait and retry, or use different service
3. **Insufficient credits:**
 - Check account balance
 - **Solution:** Add credits or use free tier option

Issue 4: JSON Output Invalid

Symptoms:

- Cannot parse JSON in downstream tools
- Missing fields

Solutions:

1. **Copy full JSON:**
 - Ensure entire content is copied
 - Use " Copy JSON" button instead of manual selection
2. **Check for truncation:**
 - Large datasets may have truncated output
 - **Solution:** Use " Download JSON" instead
3. **Validate JSON:**
 - Use online validator (jsonlint.com)
 - Check for missing brackets/commas

Issue 5: Allele Flipping Issues

Symptoms:

- Beta values seem incorrect after harmonization
- Opposite direction of effects

Causes:

1. **Palindromic SNPs:**
 - A/T or C/G alleles
 - Cannot determine strand without allele frequency
2. **Missing allele information:**
 - Harmonization cannot flip without EA/NEA

Solutions:

1. **Provide allele information:**
 - Always include ea and nea fields
2. **Check allele frequencies:**
 - Include eaf for ambiguous cases
3. **Manual review:**
 - Review harmonized output for expected directions

Error Messages Reference

Error Message	Meaning	Solution
"File is empty or has no data rows"	File has no content or only headers	Check file content
"No JSON found in response"	AI didn't return valid JSON	Retry or use different service
"Please provide an API key"	Missing API key for AI service	Enter valid API key
"Matrix dimension mismatch"	AI returned wrong-sized matrix	Retry correlation fetch
"API error: 401"	Invalid API key	Check and re-enter API key
"API error: 429"	Rate limit exceeded	Wait before retrying
"Failed to fetch from {source}"	Network/API issue	Check connection, try fallback

8 Best Practices

1. **Data Preparation:**
 - Use rsIDs for SNP identifiers
 - Include effect and non-effect alleles
 - Ensure consistent data formatting
 - Remove duplicate SNPs
2. **File Format:**
 - CSV is most reliable
 - Use UTF-8 encoding
 - Avoid special characters in headers
 - Keep file sizes reasonable (<100MB)
3. **Correlation Matrix:**
 - Use public databases when possible (free)
 - For >10 SNPs, consider AI option
 - Identity matrix is conservative fallback

- Verify correlation matrix makes biological sense
- 4. **Quality Control:**
 - Review number of harmonized SNPs
 - Check beta signs match expected directions
 - Verify correlation matrix is symmetric
 - Validate JSON structure before using
- 5. **Performance:**
 - Keep SNP count reasonable (<10,000)
 - Use GZIP compression for large files
 - Close other browser tabs for large datasets
 - Consider batch processing for very large studies

Advanced Usage

Custom JSON Manipulation

After generation, you can manually edit the JSON for advanced use cases:

```
javascript
```

```
// Example: Filter SNPs by p-value threshold
```

```
const data = JSON.parse(outputJSON);
```

```
const pvalThreshold = 5e-8;
```

```
// Calculate p-values (requires additional Z-score calculation)
```

```
const filteredIndices = data.betaX.map((beta, i) => {
```

```
  const z = beta / data.betaXse[i];
```

```
  const p = 2 * (1 - normalCDF(Math.abs(z)));
```

```
  return p < pvalThreshold ? i : null;
```

```
}).filter(i => i !== null);
```

```
// Filter all arrays
```

```
const filtered = {
```

```
  snps: filteredIndices.map(i => data.snps[i]),
```

```
  betaX: filteredIndices.map(i => data.betaX[i]),
```

```
  betaY: filteredIndices.map(i => data.betaY[i]),
```

```
  betaXse: filteredIndices.map(i => data.betaXse[i]),
```

```
  betaYse: filteredIndices.map(i => data.betaYse[i]),
```

```
  correlation: filterMatrix(data.correlation, filteredIndices),
```

```
  // ... rest of fields
```

```
};
```

Integration with R

```
r
```

```
# Load JSON in R
```

```
library(jsonlite)
```

```
# Read from file
```

```
mr_data <- fromJSON("mr_harmonized_data.json")
```

```

# Access data
snps <- mr_data$snps
betaX <- mr_data$betaX
betaY <- mr_data$betaY
correlation <- mr_data$correlation

# Use with TwoSampleMR or MendelianRandomization packages
Integration with Python
python
import json
import numpy as np

# Load JSON
with open('mr_harmonized_data.json', 'r') as f:
    mr_data = json.load(f)

# Convert to numpy arrays
beta_x = np.array(mr_data['betaX'])
beta_y = np.array(mr_data['betaY'])
correlation = np.array(mr_data['correlation'])

# Use with scipy, statsmodels, or custom MR implementation

```

9 Frequently Asked Questions

General

Q: Is this tool free to use?

A: Yes, the tool itself is completely free. Public correlation databases (LDlink, 1000 Genomes) are also free. AI services may have free tiers (DeepSeek) or require payment (Gemini).

Q: Do I need to install anything?

A: No, it runs entirely in your web browser. Just open the HTML file or access the hosted version.

Q: Is my data secure?

A: Data processing happens entirely in your browser. Files are not uploaded to any server (except for correlation matrix API calls, which only send SNP IDs).

Data & Files

Q: What's the maximum file size?

A: Browser-dependent, typically 100-500 MB. Performance may degrade with very large files.

Q: Can I use summary statistics from different populations?

A: Yes, but be cautious. Different LD structures may require population-specific correlation matrices. LDlink supports multiple populations (CEU, YRI, etc.).

Q: What if my files have different column names?

A: The tool automatically recognizes many common variants (see Field Normalization section). If issues persist, standardize to: SNP, ea, nea, beta, se, eaf.

Harmonization

Q: Why are some SNPs excluded during harmonization?

A: Common reasons:

- SNP not present in both datasets
- Missing required fields (beta, se)
- Invalid data values

Q: How does allele flipping work?

A: If effect allele in exposure = non-effect allele in outcome (and vice versa), the outcome beta sign is flipped to align with the exposure.

Q: What about palindromic SNPs (A/T, C/G)?

A: Currently not specifically handled. Include allele frequencies to help with strand alignment.

Correlation Matrix

Q: Which correlation source should I choose?

A:

- **LDlink (recommended):** Reliable, free, well-established
- **AI-generated:** For >10 SNPs or when LDlink unavailable
- **Identity matrix:** Conservative fallback, assumes no LD

Q: What's the difference between R^2 and correlation (r)?

A: $R^2 = r^2$. LDlink returns R^2 , which the tool converts to correlation using $r = \sqrt{R^2}$.

Q: Can I provide my own correlation matrix?

A: Not directly in the UI, but you can manually edit the generated JSON to replace the correlation field.

Output & Export

Q: How do I use the JSON output?

A: Import into R (jsonlite), Python (json), or other MR analysis software. The format is compatible with common MR packages.

Q: Can I re-harmonize with different settings?

A: Yes, change correlation source and click "Harmonize" again. Previous results will be overwritten.

Q: What format is the timestamp?

A: ISO 8601 format (e.g., 2025-01-15T10:30:00.000Z).

10 Others

Citation

If you use this tool in your research, please cite:

Zhang WJ. 2025. 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)

Support & Contact

For issues, questions, or feature requests:

- **Documentation:** This guide
- **Publication:** See citation above for detailed methodology
- **Email:** Contact corresponding author through journal

Version: 1.0

Last Updated: January 2025

License: Check with IAEES for usage terms

References

- Bowden J, Smith GD, Burgess S. 2015. Mendelian randomization with invalid instruments: effect estimation and bias detection through Egger regression. *International Journal of Epidemiology*, 44: 512-525. doi: <https://doi.org/10.1093/ije/dyv080>
- 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: <https://doi.org/10.1002/gepi.21758>
- del Greco F, Minelli C, Sheehan NA, Thompson JR. 2015. Detecting pleiotropy in Mendelian randomisation studies with summary data and a continuous outcome. *Stat Med*, 34(21): 2926-2940. doi: <https://doi.org/10.1002/sim.6522>
- Grant AJ, Burgess S. 2020. Pleiotropy robust methods for multivariable Mendelian randomization. arXiv, 2008.11997
- 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: <https://doi.org/10.1093/ije/dyx102>
- Lin Z, Xue H, Pan W. 2023. Robust multivariable Mendelian randomization based on constrained maximum likelihood. *The American Journal of Human Genetics*, 110(4): 592-605. doi: <https://doi.org/10.1016/j.ajhg.2023.02.014>
- Wang XT. 2023. Briefly Describe The Common Designs of Mendelian Randomization Analysis. https://mp.weixin.qq.com/s/iY4LXS4Rg_D7Y3tVd5xNTg
- Xu S, Wang P, Fung WK, Liu Z. 2023. A novel penalized inverse-variance weighted estimator for Mendelian Randomization with applications to COVID-19 outcomes. *Biometrics*, 79(3): 2184-2195. doi: <https://doi.org/10.1111/biom.13732>
- Zhang WJ. 2021a. A statistical simulation method for causality inference of Boolean variables. *Network Biology*, 11(4): 263-273. [http://www.iaees.org/publications/journals/nb/articles/2021-11\(4\)/a-method-for-causality-inference-of-Boolean-variables.pdf](http://www.iaees.org/publications/journals/nb/articles/2021-11(4)/a-method-for-causality-inference-of-Boolean-variables.pdf)
- Zhang WJ. 2021b. Causality inference of linearly correlated variables: The statistical simulation and regression method. *Computational Ecology and Software*, 11(4): 154-161. [http://www.iaees.org/publications/journals/ces/articles/2021-11\(4\)/causality-inference-of-linearly-correlated-variables.pdf](http://www.iaees.org/publications/journals/ces/articles/2021-11(4)/causality-inference-of-linearly-correlated-variables.pdf)
- Zhang WJ. 2021c. Causality inference of nominal variables: A statistical simulation method. *Computational Ecology and Software*, 11(4): 142-153. [http://www.iaees.org/publications/journals/ces/articles/2021-11\(4\)/causality-inference-of-nominal-variables-with-statistical-simulation-method.pdf](http://www.iaees.org/publications/journals/ces/articles/2021-11(4)/causality-inference-of-nominal-variables-with-statistical-simulation-method.pdf)
- Zhang WJ. 2025. 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)